

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	6
1.1 Описание программируемой системы	6
1.2 Обзор существующих решений	8
1.3 Требования к программируемой системе	10
2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ	12
2.1 Диаграмма состояний системы	12
2.2 Диаграмма классов системы	12
2.3 Диаграмма последовательности системы	13
3 ПРАКТИЧЕСКАЯ ЧАСТЬ	14
3.1 Реализация требований к системе	15
3.2 Функциональное тестирование программного продукта	20
3.3 Инструкция по эксплуатации	25
ЗАКЛЮЧЕНИЕ	26
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	27
ПРИЛОЖЕНИЯ.....	29

# ВВЕДЕНИЕ

В современной индустрии разработки ПО разработчики должны обладать не только глубокими знаниями в своей специализации, но и быть осведомлены о различных смежных технологиях. Эта работа направлена на изучение современных подходов к разработке ПО, ориентированных на решение конкретных задач и тематику исследования. Помимо приобретения практических навыков, такая деятельность формирует новое понимание традиционных задач в области разработки программных систем и приложений.

**Цель работы:** разработка программного продукта для решения квадратных уравнений с заданными коэффициентами «Решатель» на языке C++.

## **Задачи работы**

1. Описание разрабатываемой программной системы.
2. Обзор существующих решений-аналогов по данной, либо смежной теме.
3. Формулировка требований к программируемой системе.
4. Спроектировать диаграмму состояний системы.
5. Спроектировать диаграмму классов системы.
6. Спроектировать диаграмму последовательности для системы.
7. Реализовать программный продукт в соответствии с требованиями.
8. Провести функциональное тестирование программного продукта.
9. Составить инструкцию по использованию программного продукта.
10. Составить отчет по работе.
11. Сдать отчет и представить его к защите.

**Объектом данного исследования** является автоматизация решения базовых математических задач на основных языках программирования.

**Предметом исследования** в данной работе является автоматизация решения квадратных уравнений с заданными коэффициентами на языке C++.

**В качестве основных методов исследования** применены анализ, синтез, сравнение и моделирование. Практическая реализация поставленной задачи соответствует основным подходам к разработке программного обеспечения.

**Информационной базой** исследования являются открытые источники, в том числе доступные в сети Интернет, а также материалы курса «Технологии индустриального программирования», доступные через систему дистанционного обучения РТУ МИРЭА.

В данном отчете будет представлен процесс разработки программного продукта, в том числе теоретический обзор области и системы, технологическое проектирование и описание системы. а также непосредственно результаты разработки.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Описание программируемой системы

Квадратное уравнение — это алгебраическое уравнение второй степени, общий вид которого:

$$ax^2 + bx + c = 0. \#(1.1)$$

В зависимости от значений, которые могут принимать коэффициенты квадратного уравнения, можно выделить восемь типов квадратных уравнений:

$0 = 0$ . Все три коэффициента равны нулю. Уравнение верно при любом  $x$ .

$c = 0$ . Задан только коэффициент  $c$ . Такое уравнение не имеет корней и является неверным.

$ax^2 + bx + c = 0$ . Это классическое квадратное уравнение, имеющее от нуля до двух действительных корней.

$ax^2 + bx = 0$ . Левая часть уравнения раскладывается в вид  $x(ax + b) = 0$ , т. е. уравнение имеет два корня: 0 и  $-\frac{b}{a}$ .

$ax^2 + c = 0$ . Уравнение имеет корни  $\pm\sqrt{-\frac{c}{a}}$  при  $-\frac{c}{a} \geq 0$ .

$ax^2 = 0$ . Коэффициенты  $b$  и  $c$  равны нулю. Единственным корнем уравнения является 0.

$bx + c = 0$ . При  $a = 0$  уравнение является уравнением первой степени с единственным корнем  $-\frac{c}{b}$ .

$bx = 0$ . По аналогии с предыдущим типом это уравнение первой степени, но коэффициент  $c$  здесь равен нулю. Единственным корнем уравнения является 0.

Для решения квадратных уравнений используются различные методы. Выбор метода зависит от типа квадратного уравнения.

К основным методам решения квадратных уравнений обычно относят следующие [1]:

Выделение полного квадрата. Предполагается применение формулы квадрата суммы  $a^2 + 2ab + b^2 = (a + b)^2$ , а также других формул сокращённого умножения.

Разложение на множители. Используется утверждение о том, что произведение множителей равно нулю, если хотя бы один из множителей равен нулю.

Решение с помощью дискриминанта. Применяется формула для вычисления дискриминанта  $D = b^2 - 4ac$ . В зависимости от значения дискриминанта определяется количество действительных 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 не имеет действительных корней (есть комплексные). Корни уравнения можно вычислить по формуле (1.2):

$$\frac{-b \pm \sqrt{D}}{2a}. \#(1.2)$$

Теорема Виета. Согласно этой теореме, сумма корней приведённого квадратного уравнения  $x^2 + px + q = 0$  равна второму коэффициенту, взятому с противоположным знаком, а произведение корней равно свободному члену. В более простом виде формулами 1.3 это можно записать так:

$$\{x_1 + x_2 = -p, x_1 x_2 = q\}. \#(1.3)$$

Есть и другие, менее популярные, но рабочие методы решения квадратных уравнений.

Например, если для квадратного уравнения  $ax^2 + bx + c = 0$  выполняется условие  $a + b + c = 0$ , то корнями являются  $-1$  и  $\frac{c}{a}$  [2].

Используя ресурсы вычислительной машины, можно находить корни квадратных уравнений методом подбора. Для этого нужно определить интервал, в котором будет выполняться поиск корня, установить шаг и проверять все значения многочлена уравнения в данном интервале с установленным шагом. Для человека такая задача является более трудоёмкой и ресурсозатратной, чем для компьютера.

Разработана программа для автоматизации решения квадратных уравнений, поддерживающая различные типы уравнений. Программа выбирает подходящий метод решения в зависимости от коэффициентов и типа уравнения. Пользователь вводит коэффициенты  $a$ ,  $b$  и  $c$  уравнения (1.1), после чего программа находит и отображает решения в удобном формате. Для поиска корней используются различные методы. Эта программа упрощает и ускоряет процесс решения квадратных уравнений.

## 1.2 Обзор существующих решений

С появлением и развитием средств вычислительной техники решались различные математические задачи, в том числе и задача по нахождению корней квадратного уравнения. Рассмотрено три программных продукта, функционал которых включает в себя решение квадратных уравнений.

### **Mathcad**

Mathcad — это инженерное математическое программное обеспечение для выполнения, анализа инженерных расчётов и обмена ими [3].

Mathcad имеет много преимуществ, среди которых [4]:

- удобный многооконный интерфейс;
- простота использования и подробная справка;
- возможность работы без доступа к интернету;
- сильное сходство записи задач с тем, как они решаются на бумаге;

- возможность решать задачи не только по математике, но и в других сферах, таких как физика, экономика, строительство.

Среди недостатков Mathcad [4]:

- отсутствие онлайн-версии;
- возможность возникновения сложностей при установке и настройке программы;
- запрет внесения изменений в программу и работа строго по шаблону;
- отсутствие доступа для обычных пользователей (при скачивании требуется указать данные по учёбе или работе).

### **Photomath**

Photomath — это приложение, которое помогает учиться решать математические задания. Задания можно вводить как вручную, так и с помощью камеры [5].

Достоинствами приложения Photomath являются [5]:

- удобный интерфейс;
- возможность ввода заданий с помощью камеры;
- возможность работы без доступа к интернету;
- поэтапное решение заданий;
- сохранение истории при вводе заданий;
- способность решать огромное множество заданий за доли секунды;
- практически полное отсутствие ошибок при решении.

Из недостатков можно выделить следующие [5]:

- приложение доступно только как мобильное;
- приложение не всегда распознаёт написанное задание;
- есть определенные ограничения в решениях (например, приложение не воспринимает текстовые условия).

### **Microsoft Math Solver**

Microsoft Math Solver — это ещё один решатель математических задач, позволяющий получать пошаговые объяснения, создавать графическое

представление математических задач, получать различные справки и пояснения по математике на своём языке [6].

Достоинствами Microsoft Math Solver являются [7, 8]:

- пошаговые объяснения, интерактивные графики и рабочие листы;
- сохранение истории решения примеров;
- доступность как в интернете, так и в виде мобильного приложения;
- возможность ввода заданий с помощью камеры.

Среди недостатков данного решения [8]:

- отсутствие настольной версии;
- необходимость подключения к интернету;
- плохая реализация функции рисования формулы.

### 1.3 Требования к программируемой системе

В таблице 1.1 представлены требования к программируемой системе.

Таблица 1.1 – Требования к программируемой системе

№	Требование	Значение
1	Язык программирования	C++
2	Корректность работы	Приложение запускается и поддерживает стабильный цикл работы от момента старта до завершения
3	Применение принципов объектно-ориентированного программирования	При написании приложения, как минимум, были использованы классы в C++, объектный подход к проектированию системы, а также инкапсуляция.
4	Интерфейс пользователя	Создан интерфейс пользователя, поддерживающий корректный пользовательский опыт и содержащий все необходимые пояснения к работе и эксплуатации
5	Инструкция по эксплуатации	Написана инструкция по эксплуатации, содержащая, в том числе, основные рекомендации по использованию и пояснения к возможным ошибкам в программе
6	Обработка исключений	Приложение должно осуществлять проверку входных данных на корректность и целостность, а также обрабатывать возможные ошибки ввода пользователем.
7	Тестирование	Приложение протестировано на различных входных данных.



8	Использование различных методов решения уравнений	Приложение должно содержать реализации всех 8 методов решения уравнений второй степени в соответствии с требованиями задачи.
9	Обработка комплексных корней	В данном приложении предусмотрено вычисление специфического вида корней квадратного уравнения, содержащих как действительную, так и мнимую части.
10	Документация по теме “Квадратные уравнения”	Приложение может предоставить краткую документацию по типам квадратных уравнений и методам их решений.

## 2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

### 2.1 Диаграмма состояний системы

На рисунке 2.1 представлена диаграмма состояний системы.

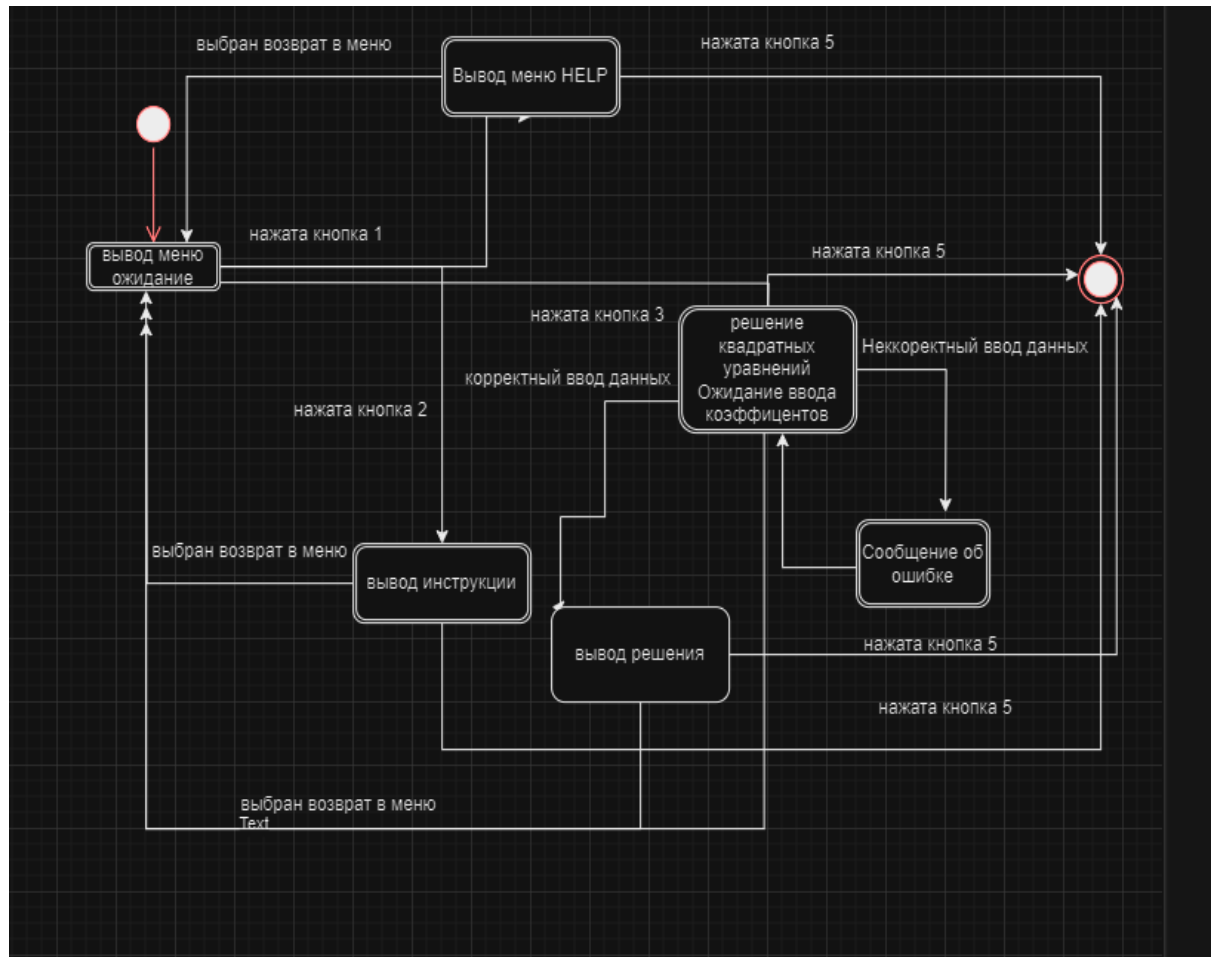


Рисунок 2.1 — Диаграмма состояний

### 2.2 Диаграмма классов системы

На рисунке 2.2 представлена диаграмма классов системы.

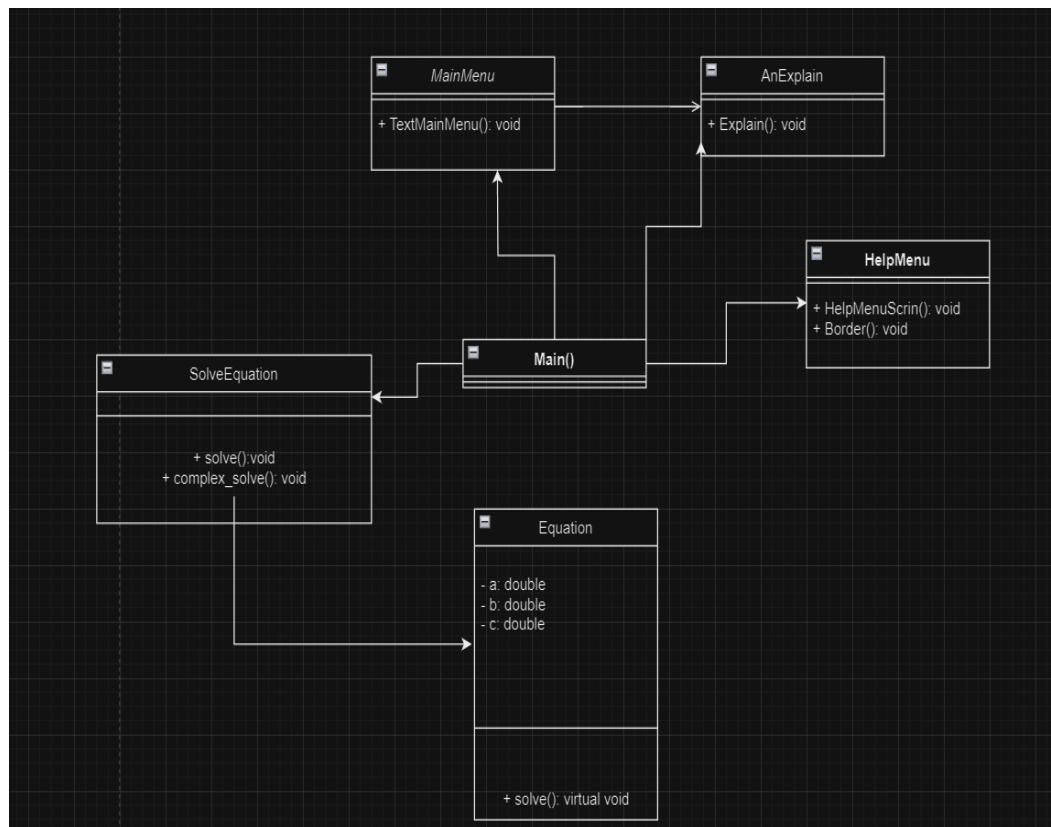
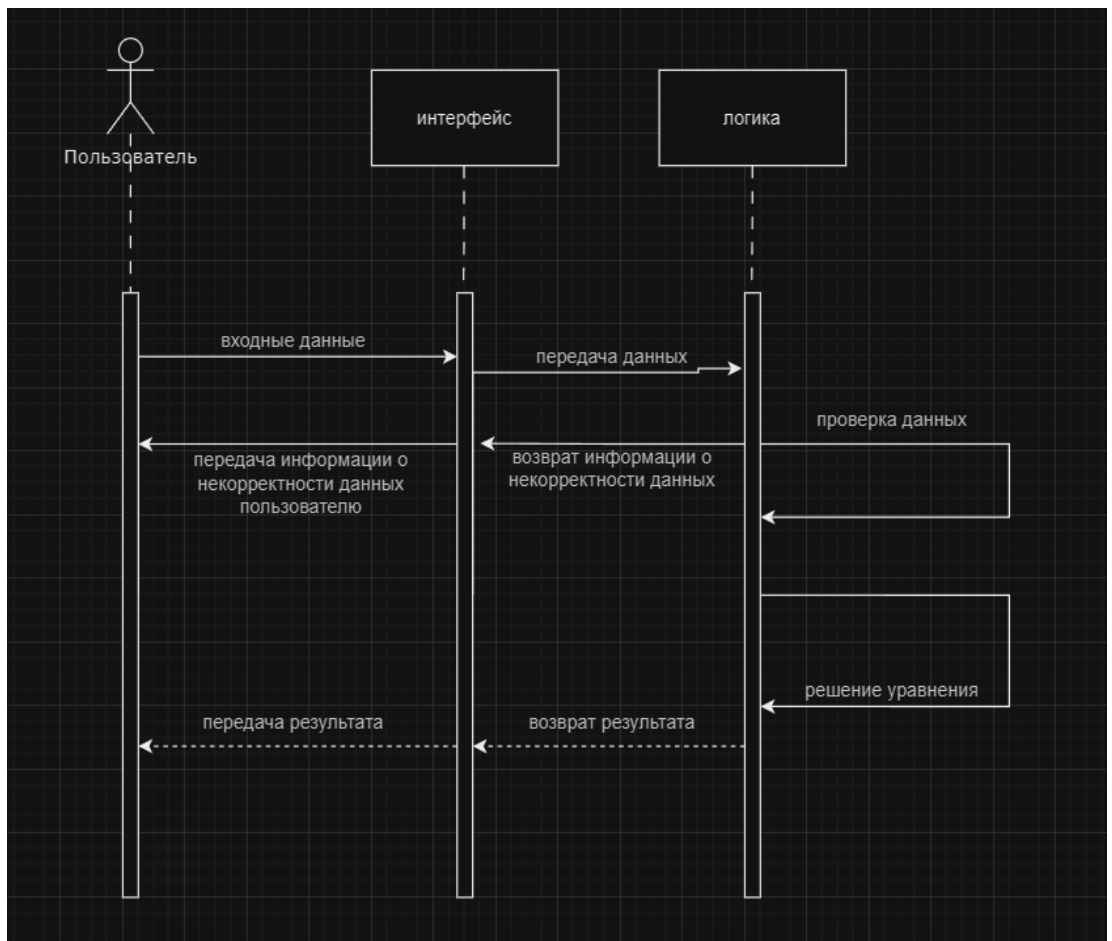


Рисунок 2.2 — Диаграмма классов

## 2.3 Диаграмма последовательности системы

На рисунке 2.3 представлена диаграмма последовательности системы.



**Рисунок 2.3 — Диаграмма последовательности**

## 3 ПРАКТИЧЕСКАЯ ЧАСТЬ

### 3.1 Реализация требований к системе

Язык программирования C++. Подтверждается на скриншотах к следующим пунктам.

Корректность работы. Проиллюстрировано в функциональном тестировании программного продукта.

Применение принципов объектно-ориентированного программирования (Рисунки 3.1.1)

```
class Equation{
protected:
    double a, b, c;
public:
    Equation(double a, double b, double c) {
        this->a = a;
        this->b = b;
        this->c = c;
    }
    virtual void solve() = 0;
};

// Solve equation class

class SolveEquation: public Equation{
public:
    SolveEquation(double a, double b, double c) : Equation(a, b, c) {}

    void solve() override {
        cout << "\033[2J\033[1;1H";
        try{
            if (a == b && b == c && b == 0){
                throw invalid_argument("There is no such equation");
            }
            if (a != 0 && c == b && b == 0){
                cout << "This quadratic equation has only one solution, x = : 0" << endl;
            }
            if (a == 0 && c == a && b != 0){
                cout << "This quadratic equation has only one solution, x = : 0" << endl;
            }
        }
    }
};
```

Рисунок 3.1.1

Используются классы, наследование, полиморфизм.

Интерфейс пользователя (Рисунки 3.1.2-3.1.3).

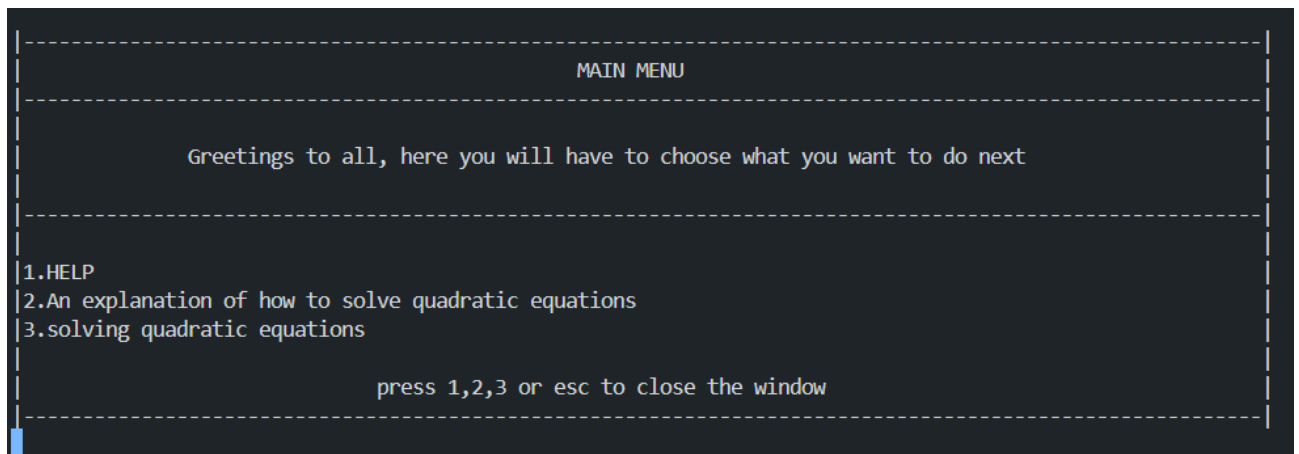


Рисунок 3.1.2

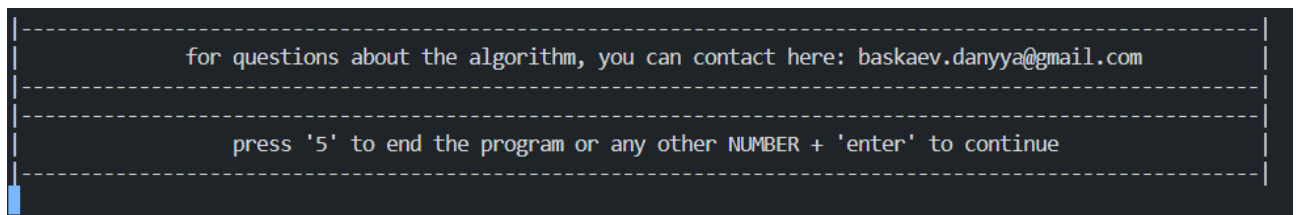


Рисунок 3.1.3

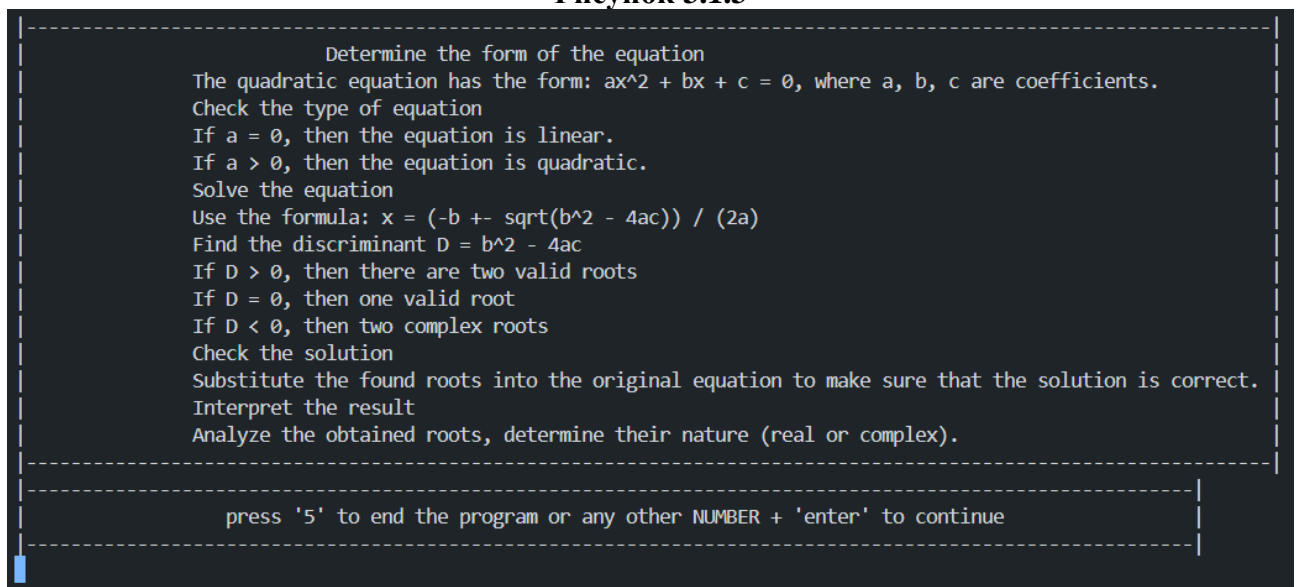


Рисунок 3.1.4

Понятный и удобный интерфейс, содержащий все необходимые сведения.

## Обработка ошибок и исключений

```
try{
    if (a == b && b == c && b == 0){
        throw invalid_argument("There is no such equation");
    }
    if (a != 0 && c == b && b == 0){
        cout << "This quadratic equation has only one solution, x = : 0" << endl;
    }
    if (a == 0 && c == a && b != 0){
        cout << "This quadratic equation has only one solution, x = : 0" << endl;
    }
    if (a == 0 && b != 0 && c != 0){
        cout << "This quadratic equation has only one solution, x = : " << (-c / b) << endl;
    }
    double discriminant = b * b - 4 * a * c;
    if (discriminant < 0 && a != 0) { // if discriminant < 0, calling the function complex_solve, which will
        complex_solve(discriminant);
    }
    else if (discriminant == 0 && a != 0 && b != 0 && c != 0) { // if discriminant = 0, this means that this
        double x = -b / (2 * a);
        cout << "This quadratic equation has only one solution, x = : " << x << endl;
    }
    if (discriminant > 0 && a != 0){
        double x1 = (-b + sqrt(discriminant)) / (2 * a);
        double x2 = (-b - sqrt(discriminant)) / (2 * a);
        cout << "This quadratic equation has two solutions:" << endl;
        cout << "x1 = " << x1 << endl;
        cout << "x2 = " << x2 << endl;
    }
}

}
catch(runtime_error& e){
    cout << "Error: " << e.what() << endl;
}
```

Рисунок 3.1.5

Рисунок 3.1.6

```
while (true) {
    if (cin >> k) {
        //right enter
        break;
    }else{
        cin.clear();
        cin.ignore();
        cout << "wrong number, try again." << endl;
    }
}
```

Рисунок 3.1.6

В коде обработаны возможные некорректные входные данные, предоставленные пользователем.

Тестирование.

Невозможно подтвердить каким-либо скриншотом. Программа была протестирована как разработчиком, так и проверяющим. Последующее тестирование предоставлено в части 3.2 отчета.

Использование различных методов решения уравнений (Рисунки 3.1.7-3.1.10).

```
if (a != 0 && c == b && b == 0){  
    cout << "This quadratic equation has only one solution, x = : 0" << endl;  
}
```

Рисунок 3.1.7

```
if (a == 0 && b != 0 && c != 0){  
    cout << "This quadratic equation has only one solution, x = : " << (-c / b) << endl;  
}
```

Рисунок 3.1.8

```
double discriminant = b * b - 4 * a * c;  
if (discriminant < 0 && a != 0) { // if discriminant < 0, calling the function complex_solve, which will be described below  
    complex_solve(discriminant);  
}  
else if (discriminant == 0 && a != 0 && b != 0 && c != 0) { // if discriminant == 0, this means that this quadratic equation has only one solution  
    double x = -b / (2 * a);  
    cout << "This quadratic equation has only one solution, x = : " << x << endl; }  
if (discriminant > 0 && a != 0){  
    double x1 = (-b + sqrt(discriminant)) / (2 * a);  
    double x2 = (-b - sqrt(discriminant)) / (2 * a);  
    cout << "This quadratic equation has two solutions:" << endl;  
    cout << "x1 = " << x1 << endl;  
    cout << "x2 = " << x2 << endl;  
}
```

Рисунок 3.1.9

```
void complex_solve(double discriminant){  
    discriminant *= -1;  
    double compl_number = sqrt(discriminant) / (2 * a);  
    cout << "This quadratic equation has a negative discriminant, so it has two complex numbers in the answer:" << endl;  
    cout << "x1 = " << -b / (2 * a) << " + " << compl_number << 'i' << endl;  
    cout << "x2 = " << -b / (2 * a) << " - " << compl_number << 'i' << endl;  
}
```

Рисунок 3.1.10

В программе определено и описано 5 классов, каждый из которых отвечает за определённую задачу, а основной класс SolveEquation, со входящими в него функциями, решает квадратные уравнения.

Обработка комплексных корней (Рисунок 3.1.11).



```
// complex number function
void complex_solve(double discriminant){
    discriminant *= -1;
    double compl_number = sqrt(discriminant) / (2 * a);
    cout << "This quadratic equation has a negative discriminant, so it has two complex numbers in the answer:" << endl;
    cout << "x1 = " << -b / (2 * a) << " + " << compl_number << 'i' << endl;
    cout << "x2 = " << -b / (2 * a) << " - " << compl_number << 'i' << endl;
}
}
```

Рисунок 3.1.11

Если дискриминант отрицательный, то программа высчитывает комплексные корни данного ей уравнения.

Документация по теме «Квадратные уравнения» (Рисунки 3.1.12-3.1.13).

```
class AnExplain{
public:
    void explain(){
        cout << "\033[2J\033[1;1H";
        cout << "|-----|" << endl;
        cout << "|               Determine the form of the equation               |" << endl;
        cout << "|       The quadratic equation has the form: ax^2 + bx + c = 0, where a, b, c are coefficients.       |" << endl;
        cout << "|       Check the type of equation                               |" << endl;
        cout << "|       If a = 0, then the equation is linear.                   |" << endl;
        cout << "|       If a > 0, then the equation is quadratic.                 |" << endl;
        cout << "|       Solve the equation                                         |" << endl;
        cout << "|       Use the formula: x = (-b +/- sqrt(b^2 - 4ac)) / (2a)      |" << endl;
        cout << "|       Find the discriminant D = b^2 - 4ac                        |" << endl;
        cout << "|       If D > 0, then there are two valid roots                  |" << endl;
        cout << "|       If D = 0, then one valid root                              |" << endl;
        cout << "|       If D < 0, then two complex roots                          |" << endl;
        cout << "|       Check the solution                                         |" << endl;
        cout << "|       Substitute the found roots into the original equation to make sure that the solution is correct. |" << endl;
        cout << "|       Interpret the result                                       |" << endl;
        cout << "|       Analyze the obtained roots, determine their nature (real or complex). |" << endl;
        cout << "|-----|" << endl;
    }
};
```

Рисунок 3.1.12

```
|-----|
|               Determine the form of the equation               |
|       The quadratic equation has the form: ax^2 + bx + c = 0, where a, b, c are coefficients.       |
|       Check the type of equation                               |
|       If a = 0, then the equation is linear.                   |
|       If a > 0, then the equation is quadratic.                 |
|       Solve the equation                                         |
|       Use the formula: x = (-b +/- sqrt(b^2 - 4ac)) / (2a)      |
|       Find the discriminant D = b^2 - 4ac                        |
|       If D > 0, then there are two valid roots                  |
|       If D = 0, then one valid root                              |
|       If D < 0, then two complex roots                          |
|       Check the solution                                         |
|       Substitute the found roots into the original equation to make sure that the solution is correct. |
|       Interpret the result                                       |
|       Analyze the obtained roots, determine their nature (real or complex). |
|-----|
|
|       press '5' to end the program or any other NUMBER + 'enter' to continue
|
|-----|
```

Рисунок 3.1.13

Пользователь имеет возможность ознакомиться с кратким теоретическим материалом по теме.

## 3.2 Функциональное тестирование программного продукта

Работа с программой начинается с вывода главного меню, в котором пользователю предоставляются опции на выбор.

(Рисунок 3.2).

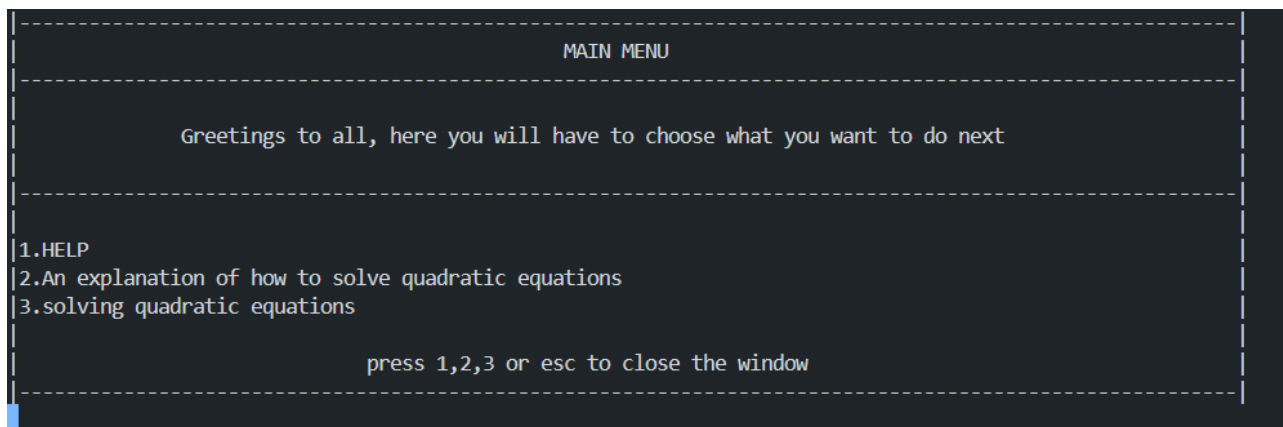


Рисунок 3.2 — Главное меню

Работа каждого из пунктов меню реализованна отдельными классами.А сам функционал данного приложения реализуется с помощью класса SolveEquation и входящими в него функциями, которые отвечают за тип квадратного уравнения.

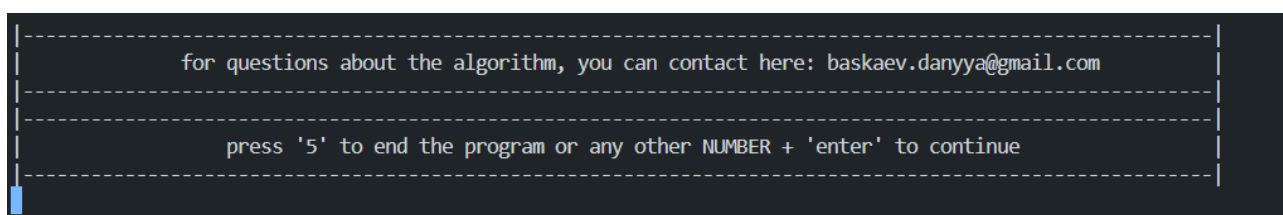


Рисунок 3.3 — HelpMenu

Класс HelpMenu отвечает за вывод вспомогательной информации в программе. В нем реализованы две функции:

1. HelpMenuScrin():

- Эта функция очищает экран и выводит сообщение с контактной информацией для получения помощи по алгоритму.

2. Border():

- Эта функция выводит границу в терминале с инструкцией о том, что нужно нажать '5' для завершения программы или любое другое число + 'Enter' для продолжения.

Эти функции используются в основной программе для предоставления пользователю необходимой информации и помощи при работе с приложением.

```
-----
                Determine the form of the equation
The quadratic equation has the form:  $ax^2 + bx + c = 0$ , where a, b, c are coefficients.
Check the type of equation
If  $a = 0$ , then the equation is linear.
If  $a > 0$ , then the equation is quadratic.
Solve the equation
Use the formula:  $x = (-b \pm \sqrt{b^2 - 4ac}) / (2a)$ 
Find the discriminant  $D = b^2 - 4ac$ 
If  $D > 0$ , then there are two valid roots
If  $D = 0$ , then one valid root
If  $D < 0$ , then two complex roots
Check the solution
Substitute the found roots into the original equation to make sure that the solution is correct.
Interpret the result
Analyze the obtained roots, determine their nature (real or complex).
-----
press '5' to end the program or any other NUMBER + 'enter' to continue
-----
```

Рисунок 3.4 — AnExplain

Класс 'AnExplain' отвечает за вывод объяснения процесса решения квадратных уравнений. В нем реализован метод 'explain()', который выводит пошаговую инструкцию по решению квадратных уравнений:

#### 1. Определение формы уравнения:

- Квадратное уравнение имеет вид ' $ax^2 + bx + c = 0$ ', где 'a', 'b', 'c' - коэффициенты.
- Если ' $a = 0$ ', то уравнение является линейным.
- Если ' $a > 0$ ', то уравнение является квадратным.

#### 2. Решение уравнения:

- Используется формула ' $x = (-b \pm \sqrt{b^2 - 4ac}) / (2a)$ '.
- Вычисляется дискриминант ' $D = b^2 - 4ac$ '.
- Если ' $D > 0$ ', то есть два действительных корня.

- Если  $D = 0$ , то есть один действительный корень.
- Если  $D < 0$ , то два комплексных корня.

### 3. Проверка решения:

- Подставляются найденные корни в исходное уравнение, чтобы убедиться в правильности решения.

### 4. Интерпретация результата:

- Анализируются полученные корни, определяется их природа (действительные или комплексные).

Таким образом, класс `AnExplain` предоставляет пользователю справочную информацию и пошаговое руководство по решению квадратных уравнений, которое может быть полезно при использовании приложения.

Рисунок 3.5 — Solve Equation

```

MAIN MENU

Greetings to all, here you will have to choose what you want to do next

1.HELP
2.An explanation of how to solve quadratic equations
3.solving quadratic equations

press 1,2,3 or esc to close the window

3
enter the coefficient a: 1
enter the coefficient b: 6
enter the coefficient c: 9

This quadratic equation has only one solution, x = :-3

press '5' to end the program or any other NUMBER + 'enter' to continue

```

Класс `SolveEquation` является подклассом (производным классом) от базового класса `Equation`. Он предназначен для решения квадратных уравнений.

Основные функции класса `SolveEquation`:

1. ``solve()``: Эта виртуальная функция переопределяет реализацию функции ``solve()`` в базовом классе ``Equation``. В ней происходит решение квадратного уравнения.

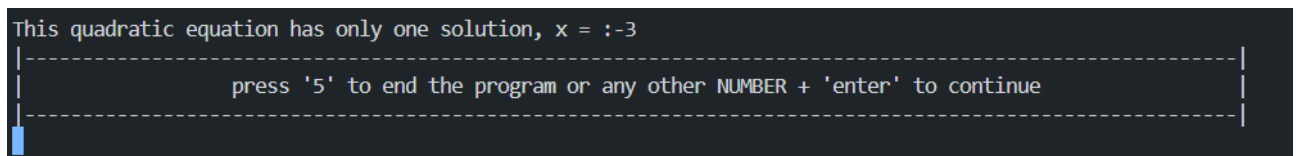
2. ``complex_solve(double discriminant)``: Эта функция вызывается, если дискриминант квадратного уравнения отрицательный. Она находит комплексные корни уравнения и выводит их на экран.

Алгоритм решения квадратного уравнения в классе ``SolveEquation``:

1. Проверяются различные особые случаи, когда коэффициенты ``a``, ``b`` или ``c`` равны нулю.
2. Вычисляется дискриминант  $D = b^2 - 4ac$ .
3. Если  $D < 0$ , вызывается функция ``complex_solve()`` для нахождения комплексных корней.
4. Если  $D = 0$ , вычисляется единственный действительный корень  $x = -b / (2a)$ .
5. Если  $D > 0$ , вычисляются два действительных корня  $x_1 = (-b + \sqrt{D}) / (2a)$  и  $x_2 = (-b - \sqrt{D}) / (2a)$ .
6. Результат решения выводится на экран.

Таким образом, класс ``SolveEquation`` инкапсулирует всю логику решения квадратных уравнений, предоставляя удобный интерфейс для работы с ними.

**Рисунок 3.6 — Выход из программы**



```
This quadratic equation has only one solution, x = :-3
-----
press '5' to end the program or any other NUMBER + 'enter' to continue
-----
```

В приложении предусмотрен выход из программы на любой стадии (рис 3.6). Так же в программе предусмотрена обработка некорректного пользовательского ввода (Рисунки 3.7-3.10).

```

his quadratic equation has only one solution, x = :-3
-----
press '5' to end the program or any other NUMBER + 'enter' to continue
-----
B
wrong number, try again.

```

**Рисунок 3.7— Обработка исключений**

```

-----
for questions about the algorithm, you can contact here: baskaev.danyya@gmail.com
-----
press '5' to end the program or any other NUMBER + 'enter' to continue
-----
B
wrong number, try again.

```

**Рисунок 3.8 — Обработка исключений**

```

-----
Determine the form of the equation
The quadratic equation has the form:  $ax^2 + bx + c = 0$ , where a, b, c are coefficients.
Check the type of equation
If a = 0, then the equation is linear.
If a > 0, then the equation is quadratic.
Solve the equation
Use the formula:  $x = (-b \pm \sqrt{b^2 - 4ac}) / (2a)$ 
Find the discriminant  $D = b^2 - 4ac$ 
If  $D > 0$ , then there are two valid roots
If  $D = 0$ , then one valid root
If  $D < 0$ , then two complex roots
Check the solution
Substitute the found roots into the original equation to make sure that the solution is correct.
Interpret the result
Analyze the obtained roots, determine their nature (real or complex).
-----
press '5' to end the program or any other NUMBER + 'enter' to continue
-----
B
wrong number, try again.

```

**Рисунок 3.9 — Обработка исключений**

```

-----
press '5' to end the program or any other NUMBER + 'enter' to continue
-----
B
wrong number, try again.

```

**Рисунок 3.10 — Обработка исключений после ввода неправильного коэффициента уравнения**

### **3.3 Инструкция по эксплуатации**

1. Настоятельно рекомендуется вводить корректные данные в соответствии с запросами программы, в противном случае будет выдано соответствующее сообщение об ошибке.
2. Перед использованием решателя квадратных уравнений рекомендуется ознакомиться с краткой теорией, чтобы лучше понять тему.
3. При работе с меню необходимо выбрать нужный пункт, ввести его номер в поле для ответа и нажать "Enter". Некорректный ввод приведет к возвращению в главное меню.
4. При работе с дробями следует использовать десятичную запись, разделяя целую и дробную части точкой.
5. Если требуется решить более одного уравнения, после получения ответа на первое уравнение вернитесь в меню и выберите пункт 3 снова.

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была описана программируемая система, рассмотрены существующие аналогичные решения, сформированы требования к системе, спроектированы диаграммы состояний, классов и последовательности системы. Сам программный продукт был разработан в соответствии с требованиями, протестирован, а также была написана инструкция по его эксплуатации.

Возможные перспективы развития данного продукта включают добавление дополнительных математических операций и интересных функций, построение графиков для уравнений, а также улучшение пользовательского интерфейса.

Практическая и исследовательская значимость данной работы заключается не только в получении опыта разработки, но и в изучении и освоении общего контекста процесса разработки программного продукта.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Давыдова, Н.Е. Различные методы решения квадратных уравнений в 8 классе / Н.Е. Давыдова // Наука и образование: отечественный и зарубежный опыт : Сборник трудов конференции Двадцать первой Международной научно-практической конференции, Белгород, 17 июня 2019 года. — Белгород: ООО ГиК, 2019. — С. 153-155. — URL: <https://elibrary.ru/item.asp?id=39195788> (дата обращения: 14.04.2023).
2. Прямостанов, С.М. Метод коэффициентов при решении квадратных уравнений / С.М. Прямостанов, Л.В. Лысогорова // Юный учёный. — 2018. — № 1.1 (15.1). — С. 66-67. — URL: <https://moluch.ru/young/archive/15/1165/> (дата обращения: 14.04.2023).
3. Озерова Г.П. Информационные технологии: Mathcad: для студентов инженерных специальностей очной и заочных форм обучения: учебно-методическое пособие / Инженерная школа ДВФУ. — Владивосток: Дальневост. федерал. ун-т, 2020. — ISBN 978-5-7444-4776-2. — URL: <https://www.dvfu.ru/upload/medialibrary/aa4/Озерова%20Г.П.%20Информационные%20технологии.%20Mathcad.pdf> (дата обращения: 14.04.2023).
4. Приложение Photomath — калькулятор онлайн в твоём телефоне // Apps4Life : [сайт]. — URL: <https://apps4.life/prilozhenie-photomath-kalkuljator-onlajn-v-tvoem-telefone/> (дата обращения: 14.04.2023).
5. Microsoft Math Solver : официальный сайт. — URL: <https://mathsolver.microsoft.com/ru> (дата обращения: 14.04.2023).
6. Яковлев, И.В. Квадратные уравнения / mathus.ru . — С.1-7. — URL: <https://mathus.ru/math/quadeq.pdf/> (дата обращения: 15.03.2024).

# ПРИЛОЖЕНИЯ

## Приложение А – Исходный код программы

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <string>

using namespace std;

class MainMenu{
public:
    //main menu function
    void TextMainMenu(){
        cout << "\033[2J\033[1;1H"; //clean terminal command
        cout << "|-----|" << endl;
        -----|" << endl;
        cout << "|                                MAIN MENU
|" << endl;
        cout << "|-----|" << endl;
        -----|" << endl;
        cout <<
        "|
|" << endl;
        cout << "|                                Greetings to all, here you will have to choose
what you want to do next                                |" << endl;
        cout << "|
|" << endl;
        cout << "|-----|" << endl;
        -----|" << endl;
        cout << "|
|" << endl;
        cout << "|1.HELP
|" << endl;
        cout << "|2.An explanation of how to solve quadratic equations
|" << endl;
        cout << "|3.solving quadratic equations
|" << endl;
        cout << "|
|" << endl;
        cout << "|                                press 1,2,3 or esc to close the
window                                |" << endl;
        cout << "|-----|" << endl;
        -----|" << endl;
    }
};

class HelpMenu{
public:
    void HelpMenuScrin(){
        cout << "\033[2J\033[1;1H";
        cout << "|-----|" << endl;
        -----|" << endl;
        cout << "|                                for questions about the algorithm, you can
contact here: baskaev.danyya@gmail.com                                |" << endl;
    }
};
```

```

        cout << "|-----|\" << endl;
    }
    //help function for border in the terminal
    void Border(){
        cout << "|-----|\" << endl;
        cout << "|                press '5' to end the program or any other
NUMBER + 'enter' to continue                |\" << endl;
        cout << "|-----|\" << endl;
    }
};

class AnExplain{
public:
    void explain(){
        cout << "\033[2J\033[1;1H";
        cout << "|-----|\" << endl;
        cout << "|                Determine the form of the
equation                |\" << endl;
        cout << "|                The quadratic equation has the form:  $ax^2 + bx$ 
+ c = 0, where a, b, c are coefficients.                |\" << endl;
        cout << "|                Check the type of equation
|\" << endl;
        cout << "|                If a = 0, then the equation is linear.
|\" << endl;
        cout << "|                If a > 0, then the equation is quadratic.
|\" << endl;
        cout << "|                Solve the equation
|\" << endl;
        cout << "|                Use the formula:  $x = (-b \pm \sqrt{b^2 - 4ac}) /$ 
(2a)                |\" << endl;
        cout << "|                Find the discriminant  $D = b^2 - 4ac$ 
|\" << endl;
        cout << "|                If  $D > 0$ , then there are two valid roots
|\" << endl;
        cout << "|                If  $D = 0$ , then one valid
root                |\" <<
endl;
        cout << "|                If  $D < 0$ , then two complex roots
|\" << endl;
        cout << "|                Check the solution
|\" << endl;
        cout << "|                Substitute the found roots into the original
equation to make sure that the solution is correct. |\" << endl;
        cout << "|                Interpret the result
|\" << endl;
        cout << "|                Analyze the obtained roots, determine their
nature (real or complex).                |\" << endl;
        cout << "|-----|\" << endl;
    }
};

// main class with constructor
class Equation{
protected:
    double a, b, c;

```

```

public:
    Equation(double a, double b, double c) {
        this->a = a;
        this->b = b;
        this->c = c;
    }
    virtual void solve() = 0;

};

// Solve equation class

class SolveEquation: public Equation{
public:
    SolveEquation(double a, double b, double c) : Equation(a, b, c) {}

    void solve() override {
        cout << "\033[2J\033[1;1H";
        try{
            if (a == b && b == c && b == 0){
                throw invalid_argument("There is no such equation or this
equation have this answer: 0 = 0");
            }
            if (a != 0 && c == b && b == 0){
                cout << "This quadratic equation has only one solution, x = : 0"
<< endl;
            }
            if (a == 0 && c == a && b != 0){
                cout << "This quadratic equation has only one solution, x = : 0"
<< endl;
            }
            if (a == 0 && b != 0 && c != 0){
                cout << "This quadratic equation has only one solution, x = : "
<< (-c / b) << endl;
            }
            double discriminant = b * b - 4 * a * c;
            if (discriminant < 0 && a != 0) { // if discriminant < 0, calling
the function complex_solve, which will be described below
                complex_solve(discriminant);
            }
            else if (discriminant == 0 && a != 0 && b != 0 && c != 0) { // if
discriminant = 0, this means that this quadratic equation has only one solution
                double x = -b / (2 * a);
                cout << "This quadratic equation has only one solution, x = : "
<< x << endl;
            }
            if (discriminant > 0 && a != 0){
                double x1 = (-b + sqrt(discriminant)) / (2 * a);
                double x2 = (-b - sqrt(discriminant)) / (2 * a);
                cout << "This quadratic equation has two solutions:" << endl;
                cout << "x1 = " << x1 << endl;
                cout << "x2 = " << x2 << endl;
            }

        }
        catch(runtime_error& e){
            cout << "Error: " << e.what() << endl;
        }
    }
}
// complex number function

```

```

void complex_solve(double discriminant){
    discriminant *= -1;
    double compl_number = sqrt(discriminant) / (2 * a);
    cout << "This quadratic equation has a negative discriminant, so it has
two complex numbers in the answer:" << endl;
    cout << "x1 = " << -b / (2 * a) << " + " << compl_number << 'i' << endl;
    cout << "x2 = " << -b / (2 * a) << " - " << compl_number << 'i' << endl;
}

};

int main() {
    double a, b, c;
    string n;
    int k;
    while (k != 5){
        MainMenu text;
        text.TextMainMenu();
        HelpMenu help;
        AnExplain exp;
        getline(cin, n);
        if (n == "1"){
            help.HelpMenuScrin();
            help.Border();
            while (true) {
                if (cin >> k) {
                    //right enter
                    break;
                }else{
                    cin.clear();
                    cin.ignore();
                    cout << "wrong number, try again." << endl;
                }
            }
        }
        if (n == "2"){
            exp.explain();
            help.Border();
            while (true) {
                if (cin >> k) {
                    //right enter
                    break;
                }else{
                    cin.clear();
                    cin.ignore();
                    cout << "wrong number, try again." << endl;
                }
            }
        }
        if (n == "3"){
            cout << "enter the coefficient a: ";
            cin >> a;
            cout << "enter the coefficient b: ";
            cin >> b;
            cout << "enter the coefficient c: ";
            cin >> c;
            try {

```

```

        SolveEquation solve(a, b, c);
        solve.solve();
    }
    catch (const runtime_error& e) {
        cout << "Error: " << e.what() << endl;
    }
    help.Border();
    while (true) {
        if (cin >> k) {
            //right enter
            break;
        }else{
            cin.clear();
            cin.ignore();
            cout << "wrong number, try again." << endl;
        }
    }
}
cout << "\033[2J\033[1;1H";
}
return 0;

```

```

}

```