Тема: Формирование детальных требований

Номер лабораторной: 1

Номер варианта: общий

Задача 1

**1. Общая постановка задачи**

Вводятся a, b, c, которые воспринимаются коэффициенты квадратного уравнения.

Найти корни уравнения.

**2. Детальные требования, тест план**

1. **Коэффициенты должны быть заданы корректно.**

1.1 a, b, c, – числа.

Если хотя бы один из них не число, сообщение *«*Ошибка, введен не численный тип данных, завершение программы*».*

1.2 a, b, c – принадлежат целым числам.

Если хотя бы одно из значений не целое число, сообщение *«*Значения должны быть заданы целыи числами, завершение программы*».*

1. **Дискриминант должен получиться корректным**

2.2 Если при извлечении корня из дискриминанта мы получаем не целое число, сообщение «Из дискриминанта не извлекается целое число!»

2.3 Если полученный дискриминант меньше нуля, сообщение «Корней нет».

**3. Уравнение может иметь и не иметь решения**

3.1 Когда все коэффициенты уравнения равны 0, сообщение: "Корней бесчисленное множество"

3.2 Если a = 0, b = 0, c не равен 0, или а не равен 0, b = 0, причём -с/b < 0 сообщение: "Корней нет"

3.3 Если a = 0, b не равен 0, а коэф. с может быть как равен, так и не равен 0, сообщение: "Корень: -c / b”

3.4 Если полученные корни уравнения совпадают, сообщение "Корень: x1”

3.5 Если найдены все корни уравнения, и они не совпадают, сообщение: "Первый корень: -sqrt(-c / a) Второй корень: sqrt(-c / a)”

**Таблица с детальными требованиями и тест планом**.

|  |  |  |  |
| --- | --- | --- | --- |
| Требование | Детальные требования | Данные | Ожидаемый результат |
| 1. **Коэффициенты должны быть заданы корректно.** | | | |
| 1.1 a, b, c, – числа | 1.1 Если хотя бы один из них не число, сообщение «Ошибка, введен не численный тип данных, завершение программы». | 1 А 6 | Сообщение: «Ошибка, введен не численный тип данных, завершение программы». |
| 1.2 a, b, c – принадлежат целым числам. | 1.2 Если хотя бы одно из значений не целое число, сообщение «Значения должны быть заданы целми числами, завершение программы». | 1 1.5 6 | Сообщение: «Значения должны быть заданы целыми числами, завершение программы». |
| 1. **Дискриминант должен получиться корректным (D – целое число, D >= 0)** | | | |
| 2.1 D является целым числом | 2.1Если при извлечении корня из дискриминанта мы получаем не целое число, сообщение «Из дискриминанта не извлекается целое число!» | 1 5 7 | Сообщение: «Из дискриминанта не извлекается целое число!» |
| 2.2 D >= 0 | 2.2 Если полученный дискриминант меньше нуля, сообщение «Корней нет». | 3 4 1 -3 | Сообщение: «Корней нет». |
| **3. Уравнение может иметь и не иметь решения** | | | |
| 3.1 a, b, c равны 0 | 3.1 Когда все коэффициенты уравнения равны 0, сообщение: "Корней бесчисленное множество" | 0 0 0 | Сообщение: "Корней бесчисленное множество" |
| 3.2 a, b равны 0, c не равен 0  Или  а не равен 0, b = 0 | 3.2 Если a = 0, b = 0, c не равен 0, или а не равен 0, b = 0, причём -с / a < 0 сообщение: "Корней нет" | 0 0 1  1 0 1 | Сообщение: "Корней нет" |
| 3.3 а = 0, b != 0, c = 0  или  a = 0, b != 0, c != 0 | 3.3 Если a = 0, b не равен 0, а коэф. с может быть как равен, так и не равен 0, сообщение: "Корень: -c / b” | 0 1 1  0 1 0 | Сообщение: "Корень: 0” |
| 3.4 Коэффициенты при решении выдают 2 одинаковых (совпавших) корня | 3.4 Если полученные корни уравнения совпадают, сообщение "Корень: x1” | 1 -2 1 | Сообщение: "Корень: -1” |
| 3.5 Найдены все корни уравнения, и они не совпадают | Если найдены все корни уравнения, и они не совпадают, сообщение: "Первый корень: -sqrt(-c / a) Второй корень: sqrt(-c / a)” | 1 5 6 | Сообщение:  “"Первый корень: -2  Второй корень: -3” |

Проект в Visual Studio

#include <iostream>

#include <cmath>

int main()

{

setlocale(LC\_ALL, "Russian"); //подключение русской раскладки

double x1 = 0, x2 = 0;

int D = 0, i = 0, k = 0;

bool ist = 0, istDOT = 0;

int a = 0, b = 0, c = 0; //значения в типе int

std::string a1 = "", b1 = "", c1 = ""; //значения в типе string

std::cin >> a1 >> b1 >> c1;

for (i = 0; i < a1.size(); i++) // проверка введеного значения на наличие букв

{

if ((a1[i] >= '0') and (a1[i] <= '9') or ((a1[i] == '-') and (i == 0)) or (a1[i] == '.'))

{

ist = 0;

if (a1[i] == '.')

{

istDOT = 1; //проверка на наличие точки в введеном значении

}

}

else

ist = 1; //флаг, показывающий наличие букв, или др. знаков, не относящихся к числам

if (ist != 0)

{

std::cout << "Ошибка, введен не численный тип данных, завершение программы";

exit(0);

}

}

if (istDOT == 1) //проверка флага на наличие точки в введеном значении

{

std::cout << "Значения должны быть заданы целми числами, завершение программы";

exit(0);

}

for (i = 0; i < b1.size(); i++) // проверка введеного значения на наличие букв

{

if ((b1[i] >= '0') and (b1[i] <= '9') or ((b1[i] == '-') and (i == 0)) or (b1[i] == '.'))

{

ist = 0;

if (b1[i] == '.')

{

istDOT = 1; //проверка на наличие точки в введеном значении

}

}

else

ist = 1; //флаг, показывающий наличие букв, или др. знаков, не относящихся к числам

if (ist != 0)

{

std::cout << "Ошибка, введен не численный тип данных, завершение программы";

exit(0);

}

}

if (istDOT == 1) //проверка флага на наличие точки в введеном значении

{

std::cout << "Значения должны быть заданы целыи числами, завершение программы";

exit(0);

}

for (i = 0; i < c1.size(); i++) // проверка введеного значения на наличие букв

{

if ((c1[i] >= '0') and (c1[i] <= '9') or ((c1[i] == '-') and (i == 0)) or (c1[i] == '.'))

{

ist = 0;

if (c1[i] == '.')

{

istDOT = 1; //проверка на наличие точки в введеном значении

}

}

else

ist = 1; //флаг, показывающий наличие букв, или др. знаков, не относящихся к числам

if (ist != 0)

{

std::cout << "Ошибка, введен не численный тип данных, завершение программы";

exit(0);

}

}

if (istDOT == 1) //проверка флага на наличие точки в введеном значении

{

std::cout << "Значения должны быть заданы целми числами, завершение программы";

exit(0);

}

if (a1[0] == '-') //проверка на ниличие минуса перед числом

k = 1;

else

k = 0;

for (i = k; i < a1.size(); i++) //превращение числа из char в int

{

a = int(a1[i]) - '0' + a \* 10;

}

if (k == 1) //при наличии минуса в string, полученное число в int умножается на -1

a \*= -1;

if (b1[0] == '-')

k = 1;

else

k = 0;

for (i = k; i < b1.size(); i++)

{

b = int(b1[i]) - '0' + b \* 10;

}

if (k == 1)

b \*= -1;

if (c1[0] == '-')

k = 1;

else

k = 0;

for (i = k; i < c1.size(); i++)

{

c = int(c1[i]) - '0' + c \* 10;

}

if (k == 1)

c \*= -1;

if (a == 0) //решение уравнения исходя из введёных коэффициентов

{

if (b == 0)

{

if(c == 0)

std::cout << "Корней бесчисленное множество";

if(c != 0)

std::cout << "Корней нет";

}

else

{

std::cout << "Корень: " << float(-c) / b;

}

exit(0);

}

if(b == 0)

{

if (float(-c) / a < 0)

{

std::cout << "Корней нет";

}

else

{

if (float(-c) / a == 0)

std::cout << "Корень: " << 0;

else

std::cout << "Первый корень: " << -sqrt(float(-c) / a) << std::endl << "Второй корень: " << sqrt(float(-c) / a);

}

exit(0);

}

D = pow(int (b), 2) - 4 \* a \* c;

if (int(sqrt(D)) != float(sqrt(D))) // проверка на извлечение целого числа из под корня

{

std::cout << "Из дискриминанта не извлекается целое число!";

exit(0);

}

if (D < 0)

{

std::cout << "Корней нет";

exit(0);

}

x1 = float(-b + sqrt(D)) / (2 \* a);

x2 = float(-b - sqrt(D)) / (2 \* a);

if (x1 == x2) // проверка на совпавшие корни

{

std::cout << "Корень: " << x1;

}

else

{

std::cout << "Первый корень: " << x1 << std::endl << "Второй корень: " << x2;

}

return 0;

}

Задача 2

1. **Общая постановка задачи**  
   Задано число ворон в стае (N). Вывести корректное предложение: «В стае N ворон», например, «В стае 1 ворона», «В стае 5 ворон», «В стае 3 вороны»
2. **Детальные требования, тест план**
3. **Число ворон в стае должно быль задано корректно.** 
   1. N – число.  
      Если N не число, то вывести сообщение «Число ворон в стае должно быть задано числом, завершение программы».
   2. N > 0. Если N <= 0, сообщение «Число ворон должно быть задано положительным числом, завершение программы».
4. **Выведенное предложение должно быть корректным.**
   1. После N, равного 1, или равного составному числительному, оканчивающемуся на один

После N, равного 1, или равного составному числительному, оканчивающемуся на один, сообщение «В стае N ворона»

* 1. После N, равного 2,3 или 4, или равного составному числительному, оканчивающемуся на 2, 3 или 4

После N, равного 2,3 или 4, или равного составному числительному, оканчивающемуся на 2, 3 или 4, сообщение «В стае N вороны»

* 1. После N, равного 5,6, 7, 8 и т. Д

После N, равного 5,6, 7, 8 и т. д., сообщение «В стае N ворон»

2.4 Если N принадлежит промежутку от 10 до 20, то сообщение «В стае N ворон»

**Таблица с детальными требованиями и тест планом**.

|  |  |  |  |
| --- | --- | --- | --- |
| Требование | Детальные требования | Данные | Ожидаемый результат |
| 1. Число ворон в стае должно быть задано корректно. (N – число) и (N> 0) | | | |
| 1.1 N - число | 1.1 Если N не число, то вывести сообщение «Число ворон в стае должно быть задано числом, завершение программы». | A | Сообщение: «Число ворон в стае должно быть задано числом, завершение программы». |
| 1.2 N> 0 | 1.2 Если N <= 0, сообщение «Число ворон должно быть задано положительным числом, завершение программы». | 0 | Сообщение: ««Число ворон должно быть задано положительным числом, завершение программы». |
| 2. Выведенное предложение должно быть корректным. (после N, равного 1, или равного составному числительному, оканчивающемуся на один), (после N, равного 2,3 или 4, или равного составному числительному, оканчивающемуся на 2, 3 или 4) и (после N, равного 5,6, 7, 8 и т. д) | | | |
| 2.1. После N, равного 1, или равного составному числительному, оканчивающемуся на один), | 2.1.После N, равного 1, или равного составному числительному, оканчивающемуся на один, сообщение «В стае N ворона» | 1 | Сообщение: «В стае 1 ворона» |
| 2.2. После N, равного 2,3 или 4, или равного составному числительному, оканчивающемуся на 2, 3 или 4 | 2.2 После N, равного 2,3 или 4, или равного составному числительному, оканчивающемуся на 2, 3 или 4, сообщение «В стае N вороны» | 3 | Сообщение «В стае 3 вороны» |
| 2.3 После N, равного 5,6, 7, 8 и т. д | 2.3 После N, равного 5,6, 7, 8 и т. д., сообщение «В стае N ворон» | 5 | Сообщение: «В стае 5 ворон» |
| 2.4 10 <= N <= 20 | 2.4 Если N принадлежит промежутку от 10 до 20, то сообщение «В стае N ворон» | 11 | Сообщение: «В стае 11 ворон» |

Проект в Visual Studio

#include <iostream>

#include <windows.h>

int main()

{

setlocale(LC\_ALL, "Russian"); //подключение русской раскладки

SetConsoleOutputCP(CP\_UTF8);

int N = 0, i = 0, k = 0;

bool ist = 0, istDOT = 0;

std::string N1 = "";

std::cin >> N1;

for (i = 0; i < N1.size(); i++) // проверка введеного значения на наличие букв

{

if ((N1[i] >= '0') and (N1[i] <= '9') or ((N1[i] == '-') and (i == 0)) or (N1[i] == '.'))

{

ist = 0;

if (N1[i] == '.')

{

istDOT = 1; //проверка на наличие точки в введеном значении

}

}

else

ist = 1; //флаг, показывающий наличие букв, или др. знаков, не относящихся к числам

if (ist != 0)

{

std::cout << "Число ворон в стае должно быть задано числом, завершение программы";

exit(0);

}

}

if (istDOT == 1) //проверка флага на наличие точки в введеном значении

{

std::cout << "Число ворон в стае должно быть задано целым числом, завершение программы";

exit(0);

}

if (N1[0] == '-') //проверка на ниличие минуса перед числом в string

k = 1;

else

k = 0;

for (i = k; i < N1.size(); i++) //превращение числа из char в int

{

N = int(N1[i]) - '0' + N \* 10;

}

if (k == 1) //при наличии минуса в типе char, полученное число в int умножается на -1

N \*= -1;

if (N <= 0) // проверка на положительные числа

{

std::cout << "Число ворон должно быть задано положительным числом, завершение программы";

exit(0);

}

if (((N >= 5) and (N <= 20)) or (N % 10 == 0) or (N % 10 == 5) or (N % 10 == 6) or (N % 10 == 7) or (N % 10 == 8) or (N % 10 == 9))

{

std::cout << "В стае " << N << " ворон";

exit(0);

}

if ((N % 10 == 2) or (N % 10 == 3) or (N % 10 == 4)) //выбор выводимого текста исходя из введёных значений

{

std::cout << "В стае " << N << " вороны";

exit(0);

}

if (N % 10 == 1)

{

std::cout << "В стае " << N << " ворона";

}

return 0;

}

Задача 3

1. **Общая постановка задачи**

Задано три числа, определяющие длины сторон треугольника. Определить тип треугольника.

1. **Детальные требования, тест план**
   1. **Треугольник должен быть задан корректно.**

1.1 a, b, c – числа.

Если хотя бы одна из сторон не число, сообщение «Стороны треугольника должны быть заданы числами, завершение программы».

* 1. a > 0, b > 0 и c > 0.

Если хотя бы одна из сторон треугольника <= 0, сообщение «Стороны треугольника должны быть заданы положительными числами, завершение программы».

* 1. Треугольник с данными сторонами должен существовать.

Если большая из сторон больше суммы двух других сторон или если большая из сторон меньше разности двух сторон, сообщение «Треугольник с данными сторонами не существует, завершение программы».

1. **Определить тип треугольника.**
   1. a^2 + b^2 = c^2. Если условие выполняется, сообщение «Треугольник прямоугольный»
   2. a^2 + b^2 < c^2. Если условие выполняется, сообщение «Треугольник тупоугольный»
   3. a^2 + b^2 > c^2. Если условие выполняется, сообщение «Треугольник остроугольный»
   4. a = b, и b = c, a = c. Если условие выполняется, сообщение «Треугольник равносторонний»
   5. a = b, или b = c, или a = c. Если условие выполняется, сообщение «Треугольник равнобедренный»

**Таблица с детальными требованиями и тест планом**.

|  |  |  |  |
| --- | --- | --- | --- |
| Требование | Детальные требования | Данные | Ожидаемый результат |
| 1. Треугольник должен быть задан корректно (a, b, c – числа), (a> 0, b> 0 и c> 0) и (треугольник с данными сторонами должен существовать) | | | |
| 1.1. a, b, c – числа | 1.1 Если хотя бы одна из сторон не число, сообщение «Стороны треугольника должны быть заданы числами, завершение программы». | 2 В 3 | Сообщение: «Стороны треугольника должны быть заданы числами, завершение программы». |
| 1.2. a> 0, b> 0 и c> 0 | 1.2 Если хотя бы одна из сторон треугольника <= 0, сообщение «Стороны треугольника должны быть заданы положительными числами, завершение программы». | 0 2 1 | Сообщение: «Стороны треугольника должны быть заданы положительными числами, завершение программы». |
| 1.3. Треугольник с данными сторонами должен существовать | 1.3. Если большая из сторон больше суммы двух других сторон или если большая из сторон меньше разности двух сторон, сообщение «Треугольник с данными сторонами не существует, завершение программы». | 5 1 2 | Сообщение: «Треугольник с данными сторонами не существует, завершение программы». |
| 2. Определить тип треугольника. | | | |
| 2.1 a^2 + b^2 = c^2 | 2.1 Если условие выполняется, сообщение «Треугольник прямоугольный» | 3 4 5 | Сообщение: «Треугольник прямоугольный» |
| 2.2 a^2 + b^2 <c^2 | 2.2 Если условие выполняется, сообщение «Треугольник тупоугольный» | 2 4 5 | Сообщение: «Треугольник тупоугольный» |
| 2.3. a^2 + b^2> c^2 | 2.3. Если условие выполняется, сообщение «Треугольник остроугольный» | 6 8 9 | Сообщение: «Треугольник остроугольный» |
| 2.4 a = b, и b = c, a = c | 2.4 . Если условие выполняется, сообщение «Треугольник … и равносторонний» | 5 5 5 | Сообщение: «Треугольник … и равнобедренный» |
| 2.5 a = b, или b = c, или a = c. | Если условие выполняется, сообщение «Треугольник … и равнобедренный» | 2 2 3 | Сообщение: «Треугольник … и равнобедренный» |

Проект в Visual Studio

#include <iostream>

#include <cmath>

#include <windows.h>

int main()

{

setlocale(LC\_ALL, "Russian"); //подключение русской раскладки

SetConsoleOutputCP(CP\_UTF8);

int N = 0, i = 0, k = 0, a = 0, b = 0, c = 0, cmax = 0, amin = 0, bsr = 0;

bool ist = 0, istDOT = 0;

std::string A1 = "", B1 = "", C1 = ""; //стороны в типе string

std::cin >> A1 >> B1 >> C1;

for (i = 0; i < A1.size(); i++) // проверка введеного значения на наличие букв

{

if ((A1[i] >= '0') and (A1[i] <= '9') or ((A1[i] == '-') and (i == 0)) or (A1[i] == '.'))

{

ist = 0;

if (A1[i] == '.')

{

istDOT = 1; //проверка на наличие точки в введеном значении

}

}

else

ist = 1; //флаг, показывающий наличие букв, или др. знаков, не относящихся к числам

if (ist != 0)

{

std::cout << "Стороны треугольника должны быть заданы числами, завершение программы";

exit(0);

}

}

if (istDOT == 1) //проверка флага на наличие точки в введеном значении

{

std::cout << "Стороны треугольника должны быть заданы целыи числами, завершение программы";

exit(0);

}

for (i = 0; i < B1.size(); i++) // проверка введеного значения на наличие букв

{

if ((B1[i] >= '0') and (B1[i] <= '9') or ((B1[i] == '-') and (i == 0)) or (B1[i] == '.'))

{

ist = 0;

if (B1[i] == '.')

{

istDOT = 1; //проверка на наличие точки в введеном значении

}

}

else

ist = 1; //флаг, показывающий наличие букв, или др. знаков, не относящихся к числам

if (ist != 0)

{

std::cout << "Стороны треугольника должны быть заданы числами, завершение программы";

exit(0);

}

}

if (istDOT == 1) //проверка флага на наличие точки в введеном значении

{

std::cout << "Стороны треугольника должны быть заданы целыи числами, завершение программы";

exit(0);

}

for (i = 0; i < C1.size(); i++) // проверка введеного значения на наличие букв

{

if ((C1[i] >= '0') and (C1[i] <= '9') or ((C1[i] == '-') and (i == 0)) or (C1[i] == '.'))

{

ist = 0;

if (C1[i] == '.')

{

istDOT = 1; //проверка на наличие точки в введеном значении

}

}

else

ist = 1; //флаг, показывающий наличие букв, или др. знаков, не относящихся к числам

if (ist != 0)

{

std::cout << "Стороны треугольника должны быть заданы числами, завершение программы";

exit(0);

}

}

if (istDOT == 1) //проверка флага на наличие точки в введеном значении

{

std::cout << "Стороны треугольника должны быть заданы целыи числами, завершение программы";

exit(0);

}

if (A1[0] == '-') //проверка на ниличие минуса перед числом

k = 1;

else

k = 0;

for (i = k; i < A1.size(); i++) //превращение числа из char в int

{

a = int(A1[i]) - '0' + a \* 10;

}

if (k == 1) //при наличии минуса в типе char, полученное число в int умножается на -1

a \*= -1;

if (a <= 0) // проверка на положительные стороны треугольника

{

std::cout << "Стороны треугольника должны быть заданы положительными числами, завершение программы";

exit(0);

}

if (B1[0] == '-')

k = 1;

else

k = 0;

for (i = k; i < B1.size(); i++)

{

b = int(B1[i]) - '0' + b \* 10;

}

if (k == 1)

b \*= -1;

if (b <= 0)

{

std::cout << "Стороны треугольника должны быть заданы положительными числами, завершение программы";

exit(0);

}

if (C1[0] == '-')

k = 1;

else

k = 0;

for (i = k; i < C1.size(); i++)

{

c = int(C1[i]) - '0' + c \* 10;

}

if (k == 1)

c \*= -1;

if (c <= 0)

{

std::cout << "Стороны треугольника должны быть заданы положительными числами, завершение программы";

exit(0);

}

cmax = max(c, max(a, b)); //сторона с максимальной длинной

amin = min(c, min(a, b)); //сторона с минимальной длинной

bsr = a + b + c - cmax - amin; //сторона со средней длинной

if (bsr + amin <= cmax) //проверка на существование треугольника

{

std::cout << "Треугольник c такими сторонами получить нельзя, завершение программы";

exit(0);

}

else

if ((cmax == amin) and (amin == bsr))

{

std::cout << "Треугольник равносторонний";

exit(0);

}

else

if (pow(cmax, 2) == pow(amin, 2) + pow(bsr, 2)) // проверка на прямоугольный треугольник и.т.д

std::cout << "Треугольник прямоугольный";

else

if (pow(cmax, 2) > pow(amin, 2) + pow(bsr, 2))

std::cout << "Треугольник тупоугольный";

else

if (pow(cmax, 2) < pow(amin, 2) + pow(bsr, 2))

std::cout << "Треугольник остроугольный";

if ((cmax == amin) or (amin == bsr) or (cmax == bsr))

std::cout << " и равнобедренный";

return 0;

}