МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное

учреждение высшего образования

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

(РУТ(МИИТ)

Кафедра «Вычислительные системы, сети и информационная безопасность»

ОТЧЕТ ПО ДИСЦИПЛИНЕ

«Проектная деятельность»

Направление: 10.03.01 Информационная безопасность

Профиль: Безопасность компьютерных систем

Выполнил: студент группы УИБ-112

Бакеев Руслан Эльдусович

Орлов Андрей Витальевич

Проверил:

Старший преподаватель Цыганова Наталия Алексеевна

Старший преподаватель Никольская Марина Николаевна

Москва 2021 г.

Оглавление

[1 Введение 4](#_Toc91442292)

[2 Общее задание 6](#_Toc91442293)

[3 Решение уравнения в Excel 7](#_Toc91442294)

[4 Решение уравнения с помощью метода прямого перебора 9](#_Toc91442295)

[4.1 Общее теоретическое описание метода 9](#_Toc91442296)

[4.2 Таблица имён 10](#_Toc91442297)

[4.3 Блок-схема 11](#_Toc91442298)

[4.3.1 Отладка блок-схемы 11](#_Toc91442299)

[4.4 Код вычислительной процедуры 11](#_Toc91442300)

[4.5 Результат / скриншот 12](#_Toc91442301)

[4.6 Промежуточный вывод 12](#_Toc91442302)

[5 Решение уравнения с помощью метода простых итераций 13](#_Toc91442303)

[5.1 Общее теоретическое описание метода 13](#_Toc91442304)

[5.2 Таблица имён 14](#_Toc91442305)

[5.3 Блок-схема 15](#_Toc91442306)

[5.4 Код вычислительной процедуры: 15](#_Toc91442307)

[5.5 Результат / скриншот 16](#_Toc91442308)

[5.6 Промежуточный вывод 16](#_Toc91442309)

[6 Решение уравнения двумя методами 17](#_Toc91442310)

[6.1 Общее теоретическое описание метода 17](#_Toc91442311)

[6.2 Код вычислительной процедуры 17](#_Toc91442312)

[6.3 Результат / скриншот 19](#_Toc91442313)

[6.4 Промежуточный вывод 19](#_Toc91442314)

[7 Проектирование графического интерфейса 20](#_Toc91442315)

[7.1 Comeback() 20](#_Toc91442316)

[7.1.1 Код функции 20](#_Toc91442317)

[7.1.2 Скриншот работы программы 21](#_Toc91442318)

[7.2 Clicked() 21](#_Toc91442319)

[7.2.1 Код функции 22](#_Toc91442320)

[7.2.2 Скриншот работы программы 22](#_Toc91442321)

[7.3 Iterations() 22](#_Toc91442322)

[7.3.1 Код функции 23](#_Toc91442323)

[7.3.2 Скриншоты работы программы 25](#_Toc91442324)

[7.4 Brute() 25](#_Toc91442325)

[7.4.1 Код функции 26](#_Toc91442326)

[7.4.2 Скриншоты работы программы 27](#_Toc91442327)

[7.5 Special() 28](#_Toc91442328)

[7.5.1 Код функции 29](#_Toc91442329)

[7.5.2 Скриншоты работы программы 29](#_Toc91442330)

[8 Оценка времени решения 30](#_Toc91442331)

[8.1 Замер времени выполнения программы оптимизированного прямого перебора. 30](#_Toc91442332)

[8.2 Замер времени выполнения программы методом итераций. 31](#_Toc91442333)

[9 Вывод: 32](#_Toc91442334)

# 1 Введение

Задача состоит в решении уравнения несколькими способами: модуль «Поиск решения» в Excel’e, метод итераций, метод прямого перебора значений и одновременно 2 метода – итераций и перебора (доп. задание). Ниже идут записи начала работы.

Задача состоит в решении уравнения несколькими способами: модуль «Поиск решения» в Excel’e, метод итераций, метод прямого перебора значений и одновременно 2 метода – итераций и перебора (доп. задание). Ниже идут записи начала работы.  
Методы решения задачи последующие: понять, имеет ли уравнение корни с помощью производной, затем исследовать с помощью графика Excel, далее идёт нахождение корней исходного уравнения с помощью программного кода, заключающего пару итерационных методов, далее сделать простой пользовательский интерфейс для удобного вычисления корней уравнения.  
Начали искать решение уравнения вручную, по итогу сошлись на интересных результатах – изначальное уравнение не имеет корней.  
Изначально уравнение имело вид exp(-x) + x + 2 = 0. Обозначили уравнение за функцию, ищем её минимальное значение с помощью производной: exp(-x)’ = -1\*exp(-x), x’ = 1, 2’ = 0.  
Уравнение производной по итогу приняло вид -1\*exp(-x) + 1 = 0; -exp(-x) = -1, exp(-x) = 1; exp(-x) = exp(0); -x = 0; x = 0.  
Подставляя найденную точку экстремума, определяем знаки производной, выясняем, что это точка минимума. Подставим её в изначальную функцию, получается exp(-0) + 0 + 2 -> 1+2=3.  
Получается, что минимальное значение функции оказалось < 0, из этого следует, что в рациональном мире чисел решения данного уравнения не существует, остаётся искать его в комплексном.  
Так как нахождение корней в комплексном виде выходит за рамки установленного задания, оно было заменено уравнением e^(-x) – x + 2.  
Исследование уравнения в Excel'e показало наличие корней у этого уравнения  
Далее было решено сделать пункты 2 и 3, на это ушло больше времени чем ожидалось. Если третий пункт с численным методом оказался довольно простым в реализации - просто постепенно изменяем x на выражение f(x), то со вторым получилось гораздо сложнее - и тут дело в программной реализации, эффективности.  
  
Если посчитать уравнение с точностью одна миллионная, то приходиться проходить столько же итераций. И если даже брать в расчёт, что с++ довольно быстрый язык, большими значениями мы просто убьём подобным кодом машину.  
  
Надо действовать хитрее - отбирать значения в геометрической регрессии, то бишь перебираем [-1.5; 5] на отрезок ответа (такой отрезок значений [х, y], при котором f(x) < 0, f(y) > 0) с точностью n \* (10\*\*(lovely- 1)), далее уменьшенный отрезок перебираем уже до сотых, тысячных, и так пока не дойдём до нашей заданной точности. На ней и остановимся - звучит просто, но сил ушло много на реализацию.

# Общее задание

Задано иррациональное уравнение e^(-x) – x + 2 = 0 на диапазоне [-1.5;5].

1. Найти решение уравнения на указанном диапазоне с заданной точностью, используя модуль «Поиск решения» пакета Microsoft Excel.
2. Найти решение уравнения на указанном диапазоне с заданной точностью, используя метод прямого перебора значений в пределах диапазона.
3. Найти решение уравнения на указанном диапазоне с заданной точностью, используя численный метод – Метод итераций (последовательных приближений).
4. Программа должна быть написана с применением элементов графического интерфейса пользователя (окна, кнопки, поля, редактирования, метки и т.п.)
5. Определить время вычисления по процедурам (2) и (3), сравнить результаты и пояснить их(опционально).

# 3 Решение уравнения в Excel

Задание 1: Найти решение уравнения на указанном диапазоне с заданной точностью, используя модуль «Поиск решения» пакета Microsoft Excel.

Описание: найдено решение уравнения на указанном диапазоне с заданной точностью, использовав модуль «Поиск решения». С помощью точности, которая была равна 0.1, постепенно перебирался диапазон значений, подставляя эти значения вместо «x». Также был начерчен график для того, чтобы посмотреть, как зависит функция от самих значений (если при отрицательном значении – функция положительна, то при положительном значении – функция стремится к минус бесконечности)

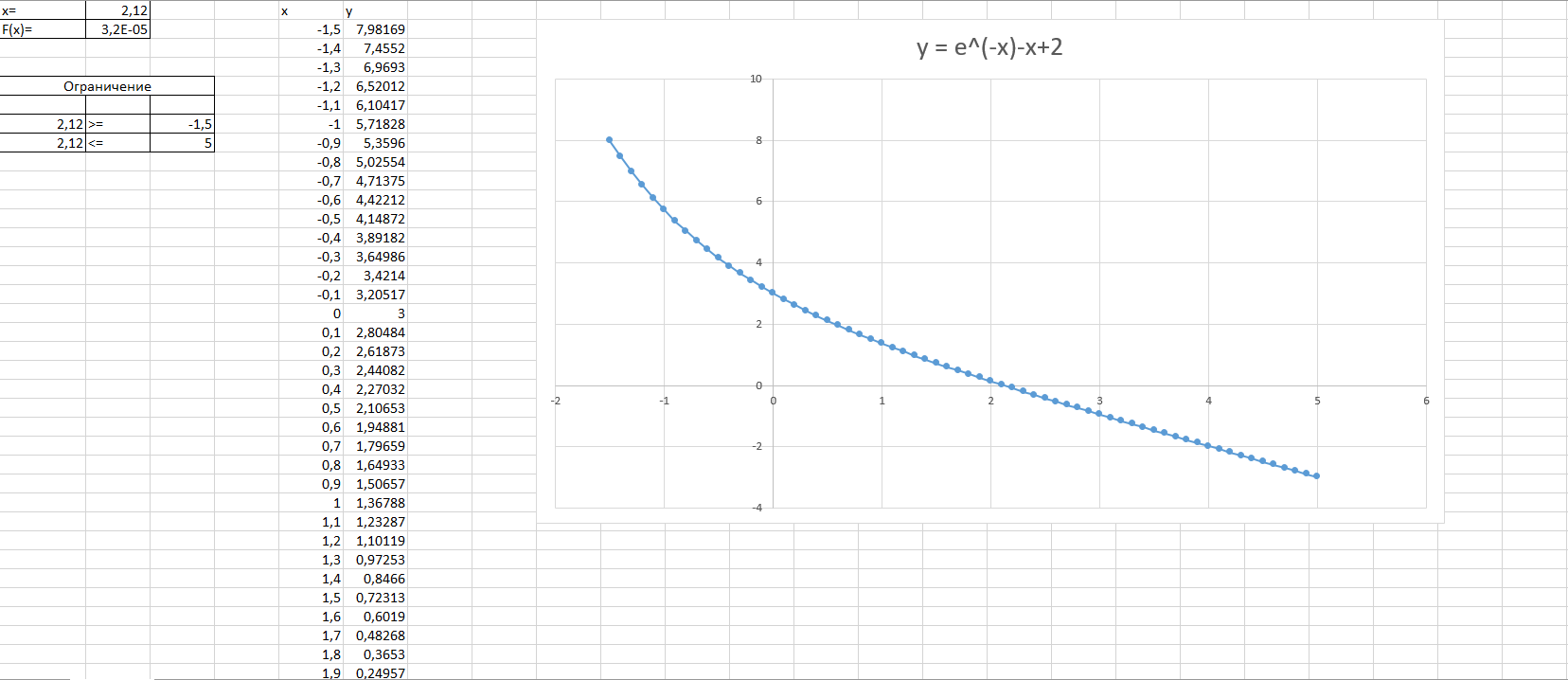


Рисунок 1 - Общий вид решения Excel

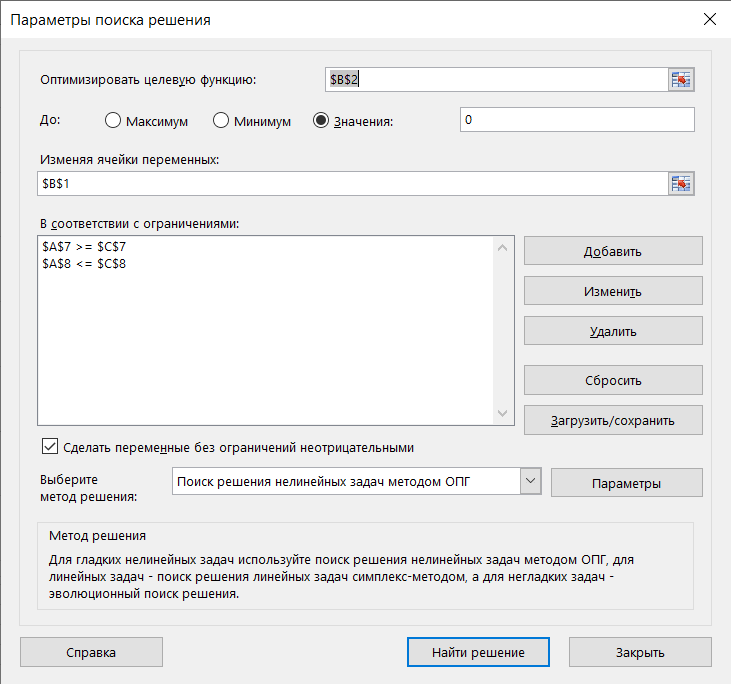


Рисунок 2 – вид модуля «Поиск решения»

Вывод: Нашли решение уравнения, изначально построив 2 таблицы: таблицу ограничений и таблицу с нахождение функции F(x). Далее изучив модуль «Поиск решения», построили график зависимости значения y от значения переменной x, которое изменялось на всём промежутке: чем меньше значения x – тем больше значение y, и наоборот.

# Решение уравнения с помощью метода прямого перебора

Задание 2: Найти решение уравнения на указанном диапазоне с заданной точностью, используя метод прямого перебора значений в пределах диапазона

# 4.1 Общее теоретическое описание метода

Алгоритм метода состоит в переборе значений на определённом отрезке, изменяющихся с заданным шагом. Функция представляет собой оптимизированный перебор на интервале. Оптимизация: вначале алгоритм ищет первоначальный отрезок, в котором лежит ответ на уравнение путём перебора отрезка на уровне десятых. Далее нужно подсчитать количество знаков после запятой в заданной точности, и углубляясь всё дальше в перебор - перебираем уже на уровне сотых, затем тысячных… Дойдя до нужного числа знаков, алгоритм останавливается, и путем деления отрезка напополам с приводом к типу данных Decimal для увеличения точности, возвращается ответ. Плюсом данного метода является простота и эффективность при нахождении точки экстремума. Но с увеличением точности и увеличивается время выполнения работы программы.

# 4.2 Таблица имён

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| n1 | Вещественный | Переменная точности |
| Рабочие переменные | | |
| fla | Логический | Флаг нахождения правой границы ответа |
| flb | Логический | Флаг нахождения левой границы ответа |
| n | Вещественный | Заданная точность |
| left | Вещественный | Левая граница отрезка ответа |
| right | Вещественный | Правая граница отрезка ответа |
| counter | Целочисленный | Кол-во знаков после запятой у числа |
| lovely | Вещественный | Переменная, копирующая заданное значение точности для анализа кол-ва знаков после запятой (подсчёта) |
| tool | Целочисленный | Переменная для постепенного углубления в найденный отрезок ответа |
| w | Целочисленный | Переменная-счётчик |
| i | Целочисленный | Переменная-счётчик |
| a | Целочисленный | Переменная-помощник правой границы ответа |
| b | Целочисленный | Переменная-помощник левой границы ответа |
| Результат | | |
| answer | Вещественный | Ответ уравнения |

Таблица 1 – Таблица имён

# 4.3 Блок-схема

Diagram

Description automatically generated

Рисунок 3 – блок-схема

# 4.3.1 Отладка блок-схемы

1. Начало

2. Ввод с клавиатуры n = 0.01

3. Объявление и инициализация переменных fla = False, flb = False, left = 0, right = 0

4. Объявление и инициализация переменной i = -15

5. i != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -1.5, fla = True

0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -14

5. -14 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -1.4, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -13

5. -13 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -1.3, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -12

5. -12 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -1.2, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -11

5. -11 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -1.1, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -10

5. -10 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -1.0, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -9

5. -9 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.9, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -8

5. -8 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.8, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -7

5. -7 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.7, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -6

5. -6 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.6, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -5

5. -5 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.5, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -4

5. -4 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.4, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -3

5. -3 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.3, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -2

5. -2 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.2, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = -1

5. -1 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = -0.1, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 0

5. 0 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 1

5. 1 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.1, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 =2

5. 2 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.2, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 3

5. 3 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.3, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 4

5. 4 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.4, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 5

5. 5 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.5, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 6

5. 6 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.6, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 7

5. 7 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.7, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 8

5. 8 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.8, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 9

5. 9 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 0.9, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 10

5. 10 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 11

5. 11 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.1, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 12

5. 12 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.2, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 13

5. 13 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.3, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 14

5. 14 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.4, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 15

5. 15 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.5, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 16

5. 16 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.6, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 17

5. 17 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.7, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 18

5. 18 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.8, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 19

5. 19 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 1.9, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 20

5. 20 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 2, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 21

5. 21 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, да, идём к блоку 7

7. Переприсваивание переменных и инициализация right = 2.1, fla = True

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 22

5. 22 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 23

5. 23 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 24

5. 24 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 25

5. 25 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 26

5. 26 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 27

5. 27 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 28

5. 28 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 29

5. 29 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 30

5. 30 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 31

5. 31 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 32

5. 32 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 33

5. 33 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 34

5. 34 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 35

5. 35 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 36

5. 36 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 37

5. 37 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 38

5. 38 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 39

5. 39 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 40

5. 40 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 41

5. 41 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 42

5. 42 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 43

5. 43 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 44

5. 44 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 45

5. 45 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 46

5. 46 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 == 47

5. 47 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 48

5. 48 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 49

5. 49 != 50, да, идём к блоку 6

6. exp(-i/10) – i/10 + 2 > 0 and fla == False, нет, идём к блоку 8

8. 0 < exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 < exp(-right) – right + 2 and fla == True, нет, идём к блоку 10

10. exp(-i/10) – i/10 + 2 < 0 and flb == False, нет, идём к блоку 12.

12. Ссылка на блок 13

13. 0 > exp(-i/10) – i/10 + 2 and exp(-i/10) – i/10 + 2 > exp(-left) – left + 2 and flb == True, нет, идём к блоку 15

15. Переприсваивание переменных i += 1 = 50

5. 50 != 50, нет, идём к блоку 16(ссылка на блок 17)

17. Объявление переменных lovely = 0.01, counter = 0

18. int(lovely) == 0, да, 0.01 == 0 идём к блоку 19

19. lovely = 0.1, counter = 1, возвращаемся в 18 блок

18. int(lovely) == 0, да 0.1 = 0 идём к блоку 19

19. lovely = 1, counter = 2, возвращаемся в 18 блок

18. int(lovely) == 0, нет 1 != 0 идём к блоку 20

20. Переприсваивание и инициализация переменных tool = 100, i = 1

21. i != counter, да, идём к блоку 22

22. присваивание переменных a = right = 2.1, b = left = 2.2

23. Переприсваивание и инициализация переменных a = 210, b = 220

24. присваивание и инициализация переменной k = a = 210

25. k != b, да 210 != 220, идём к блоку 26

26. exp(-2.2) – 2.2 + 2 < exp(-2.10) – 2.10 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.10) – 2.10 + 2 < exp(-2.1) – 2.1 + 2, нет, идём к блоку 30

30. Переприсваивание переменных k = 211, возвращаемся к блоку 25

25. k != b, да 211 != 220, идём к блоку 26

26. exp(-2.2) – 2.2 + 2 < exp(-2.11) – 2.11 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.11) – 2.11 + 2 < exp(-2.10) – 2.10 + 2, да, идём к блоку 29

29. right = k/tool = 2.11

30. Переприсваивание переменных k = 212, возвращаемся к блоку 25

25. k != b, да 212 != 220, идём к блоку 26

26. exp(-2.2) – 2.2 + 2 < exp(-2.12) – 2.12 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.12) – 2.12 + 2 < exp(-2.11) – 2.11 + 2, да, идём к блоку 29

29. right = k/tool = 2.12

30. Переприсваивание переменных k = 213, возвращаемся к блоку 25

25. k != b, да 213 != 220, идём к блоку 26

26. exp(-2.2) – 2.2 + 2 < exp(-2.13) – 2.13 + 2 < 0, да, идём к блоку 27

27. left = k/tool = 213/100 = 2.13

28. 0 < exp(-2.13) – 2.13 + 2 < exp(-2.12) – 2.12 + 2, нет, идём к блоку 30

30. Переприсваивание переменных k = 214, возвращаемся к блоку 25

25. k != b, да 214 != 220, идём к блоку 26

26. exp(-2.13) – 2.13 + 2 < exp(-2.14) – 2.14 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.14) – 2.14 + 2 < exp(-2.12) – 2.12 + 2, нет, идём к блоку 30

30. Переприсваивание переменных k = 215, возвращаемся к блоку 25

25. k != b, да 215 != 220, идём к блоку 26

26. exp(-2.13) – 2.13 + 2 < exp(-2.15) – 2.15 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.15) – 2.15 + 2 < exp(-2.12) – 2.12 + 2, нет, идём к блоку 30

30. Переприсваивание переменных k = 216, возвращаемся к блоку 25

25. k != b, да 216 != 220, идём к блоку 26

26. exp(-2.13) – 2.13 + 2 < exp(-2.16) – 2.16 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.16) – 2.16 + 2 < exp(-2.12) – 2.12 + 2, нет, идём к блоку 30

30. Переприсваивание переменных k = 217, возвращаемся к блоку 25

25. k != b, да 217 != 220, идём к блоку 26

26. exp(-2.13) – 2.13 + 2 < exp(-2.17) – 2.17 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.17) – 2.17 + 2 < exp(-2.12) – 2.12 + 2, нет, идём к блоку 30

30. Переприсваивание переменных k = 218, возвращаемся к блоку 25

25. k != b, да 218 != 220, идём к блоку 26

26. exp(-2.13) – 2.13 + 2 < exp(-2.18) – 2.18 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.18) – 2.18 + 2 < exp(-2.12) – 2.12 + 2, нет, идём к блоку 30

30. Переприсваивание переменных k = 219, возвращаемся к блоку 25

25. k != b, да 219 != 220, идём к блоку 26

26. exp(-2.13) – 2.13 + 2 < exp(-2.19) – 2.19 + 2 < 0, нет, идём к блоку 28

28. 0 < exp(-2.19) – 2.19 + 2 < exp(-2.12) – 2.12 + 2, да, идём к блоку 30

30. Переприсваивание переменных k = 220, возвращаемся к блоку 25

25. k != b, нет 220 = 220, идём к блоку 31

31. tool = 1000, i = 2, идём к блоку 21

21. i != counter, нет 2 = 2, идём к блоку 32(ссылка на блок 33)

33. answer = (left + right)/2 = (2.13 + 2.12)/2 = 2.125

34. Вывод answer = 2.125

35. Конец

4.3.2 Скриншот правдивости отладки

Text

Description automatically generated

# 4.4 Код вычислительной процедуры

def method\_of\_bruteforce(n1): //подбор отрезка на уровня десятых

def equation(x):

return exp(-x) - x + 2

n, fla, flb, left, right = n1, False, False, 0, 0

for i in range(-15, 50):

if (i / 10) > 0 and not fla:

right = i / 10

fla = True

if 0 < equation(i / 10) < equation(right) and fla:

right = i / 10

if equation(i / 10) < 0 and not flb:

left = i / 10

flb = True

if equation(left) < equation(i / 10) < 0 and flb:

left = i / 10

lovely, counter = n, 0

while int(lovely) == 0: //Подсчёт количества знаков после запятой

lovely \*= 10

counter += 1

tool = 10   
 //подбор отрезка на выбранном пользователе уровне

for i in range(counter):

a, b = right, left

a, b = int(a \* tool), int(b \* tool)

for w in range(a, b):

if equation(left) < equation(w / tool) < 0:

left = w / tool

if 0 < equation(w / tool) < equation(right):

right = w / tool

tool \*= 10  
 answer = Decimal(left + right)/2

return answer

# 4.5 Результат / скриншот

Text

Description automatically generated

Рисунок 4 – результат работы

Text

Description automatically generated

Рисунок 5 – результат работы

4.6 Промежуточный вывод: программа работает исправно с методом прямого перебора, при заданном значении пользователем (вплоть до 1E-23), выдавая конкретный ответ в окне вывода.

# 5 Решение уравнения с помощью метода простых итераций

Задание 3: Найти решение уравнения на указанном диапазоне с заданной точностью, используя численный метод – Метод итераций (последовательных приближений).

5.1 Общее теоретическое описание метода

Этот метод решения уравнений состоит в замене исходного уравнения эквивалентным ему выражением, где в левой части находится чистая переменная «x», а в правой - некое выражение, содержащее «x». Для начала найдём примерный корень уравнения, чтобы метод был гораздо точнее - для этого будем постепенно сжимать заданный отрезок. Начальный сегмент блок-схемы и кода, с громадным количеством условий, лишь шаг поиска решения, ведь он заключается в нахождении примерного ответа с целью уже заранее приблизить алгоритм к нужному числу. Число «а» постоянно растёт, в то время как «b» уходит в минус бесконечность. Это позволяет исследовать функцию на всей числовой оси Oх в целых числах. Как только находится отрезок ответа, берётся его середина и отправляется в указанное пользователем количество итераций, являющих собой вычисления ранее упомянутого выражения от найденного приближённого значения ответа, затем – подставление полученного значения обратно в выражение, вычисление его от полученного значения – и этот цикл будет длится столько раз, сколько пожелает пользователь – именно поэтому точность в программе является натуральным числом – ведь не может быть 1.5 итерации, к примеру. После всех вышеописанных процедур алгоритм выдаёт ответ.

# 5.2 Таблица имён

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| Amount\_iterations | Целочисленный | Указанная точность |
| Рабочие переменные | | |
| a | Целочисленный (1)  Вещественный (2) | 1) Переменная поиска границы отрезка ответа на положительной стороне функции  2) Середина найденного отрезка ответа заданного уравнения |
| b | Целочисленный (1)  Вещественный (2) | 1) Переменная поиска границы отрезка ответа на отрицательной стороне функции  2) Переменная изменения функции в ходе итераций последовательного приближения |
| flag\_right | Логический | Флаг факта нахождения правой границы отрезка ответа |
| flag\_left | Логический | Флаг факта нахождения левой границы отрезка ответа |
| answer | Вещественный | Переменная для подсчёта значения функции от указанного значения аргумента |
| b | Вещественный | Ответ уравнения |
| right\_edge | Вещественный | Правая граница отрезка ответа |
| left\_edge | Вещественный | Левая граница отрезка ответа |

Таблица 2 – Таблица имён

# 5.3 Блок-схема

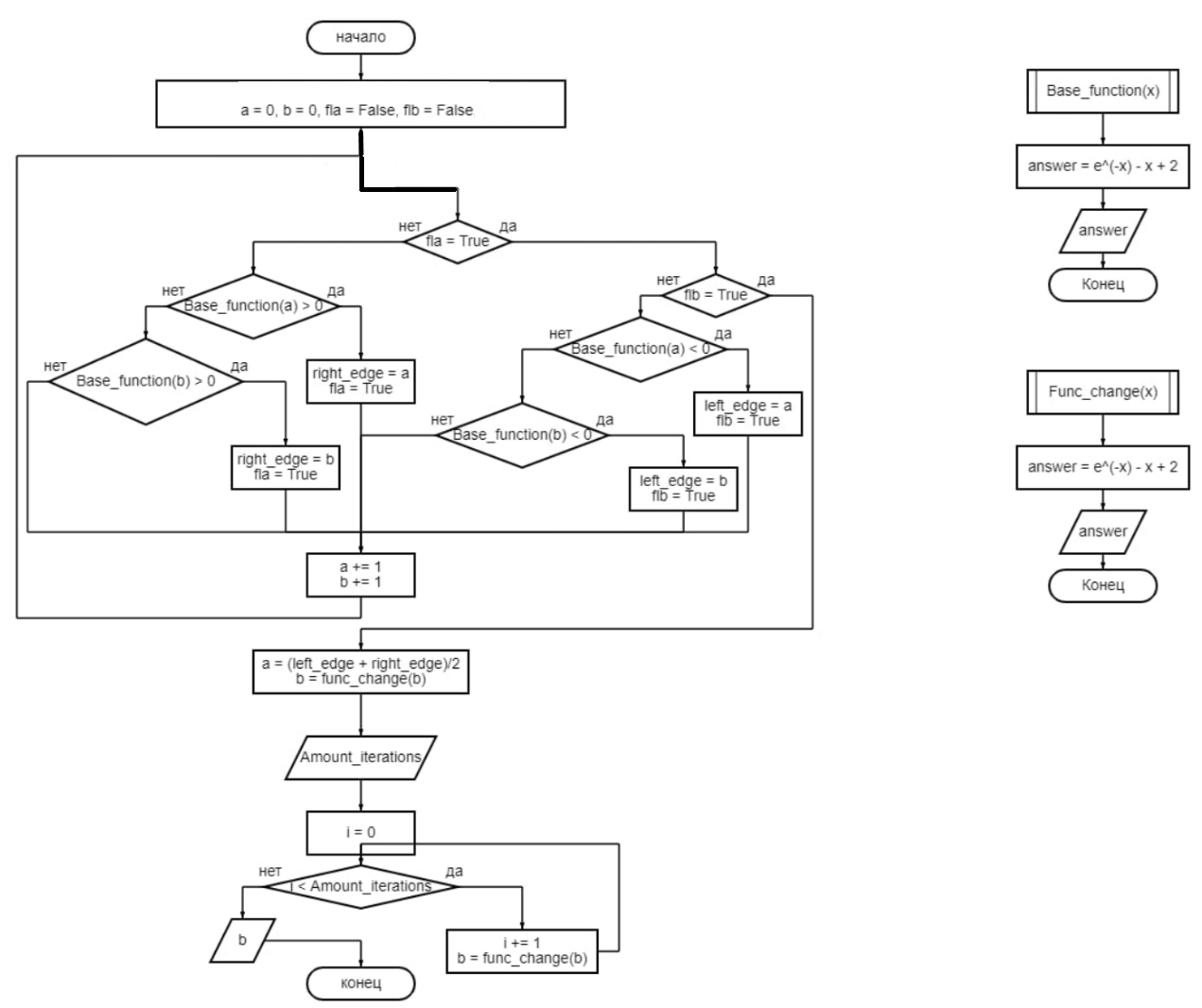


Рисунок 6 – блок-схема

# 5.4 Код вычислительной процедуры:

def method\_of\_iterations(n):

def func\_change(x): // выразили икс через изначальное уравнение

return 2 + exp(-x)

def base\_function(x): // изначальное уравнение

return exp(-x) - x + 2

amount\_iterations = n

a, b, flag\_right, flag\_left, right\_edge, left\_edge = 0, 0, False, False, 0, 0

while True:

if not flag\_right:

if base\_function(a) > 0:

right\_edge, flag\_right = a, True

if base\_function(b) > 0:

right\_edge, flag\_right = b, True

if not flag\_left:

if base\_function(a) < 0:

left\_edge, flag\_left = a, True

if base\_function(b) < 0:

left\_edge, flag\_left = b, True

a += 1

b -= 1

if flag\_right and flag\_left:

break

a = (left\_edge + right\_edge) / 2

b = func\_change(a)

for i in range(amount\_iterations):

b = func\_change(b)

return b

# 5.5 Результат / скриншот

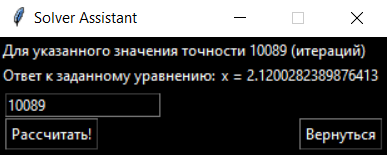


Рисунок 7 – результат работы

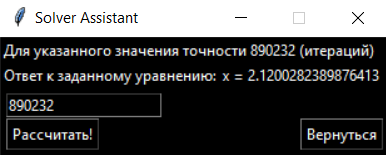


Рисунок 8 – результат работы

5.6 Промежуточный вывод: программа работает исправно с методом итераций, при заданном значении пользователем (вплоть до 1E+09), выдавая конкретный ответ в окне вывода.

# Решение уравнения двумя методами

# 6.1 Общее теоретическое описание метода

Это метод решения уравнения двумя методами: итераций и прямого перебора значений (вычисления идут параллельно).

# 6.2 Код вычислительной процедуры

# Так как этот метод по своей сути является просто сложением двух # предыдущих методов, то следует немного точнее сказать – это не КВП (код # вычислительной процедуры), а скорее КГИ (код графического интерфейса).  
# Оставим прежнее название для поддержания структуры отчёта касательно # пунктов 2 и 3.

def special():

def destroyer():

avocado.destroy()

bi.destroy()

words.destroy()

request.destroy()

something.destroy()

helped.destroy()

if please:

tr.destroy()

tr1.destroy()

comeback()

def callback(event):

global please, something

if something.get().isdigit():

if 0 < int(something.get()) < 10000001:

please = True

else:

helped.configure(text='Что же, как вводить число, вы поняли.\n'

'Но всё-таки следуйте требованиям по вводу, хорошо?', fg='red')

else:

helped.configure(text='Что же, как вводить число, вы поняли.\n'

' Но всё-таки следуйте требованиям по вводу, хорошо?', fg='red')

def strange():

global please, something

if something.get().isdigit():

if 0 < int(something.get()) < 10000001:

please = True

else:

helped.configure(text='Что же, как вводить число, вы поняли.\n'

'Но всё-таки следуйте требованиям по вводу, хорошо?', fg='red')

else:

helped.configure(text='Что же, как вводить число, вы поняли.\n'

' Но всё-таки следуйте требованиям по вводу, хорошо?', fg='red')

global please, something

bi = Button(window, text="Рассчитать!", bg='black', fg='white', command=strange)

helped = Label(window, text='(Для ввода нажмите Enter, или кнопку под полем ввода)', bg='black', fg='white')

something = Entry(window, width=15, fg='white', bg='black')

window.geometry("500x110")

lbl.destroy()

lbl1.destroy()

lbl2.destroy()

bt.destroy()

bt1.destroy()

btn2.destroy()

window.update()

avocado = Button(window, text="Вернуться", bg='black', fg='white', command=destroyer)

avocado.place(x=425, y=80)

words = Label(window, text='Вы выбрали способ нахождения решения уравнениями обоими методами.',

bg='black', fg='white')

words.place(x=5, y=2)

request = Label(window, text='Пожалуйста, укажите точность натуральным числом, до 10000000!',

bg='black', fg='white')

request.place(x=5, y=18)

helped.place(x=110, y=40)

something.place(x=10, y=47)

bi.place(x=10, y=80)

please = False

while not please:

window.update()

window.bind('<Return>', callback)

var = something.get()

mot = Decimal(method\_of\_iterations(int(var)))

mob = Decimal(method\_of\_bruteforce(1/int(var)))

tr = Label(window, text=f'Методом итерации получено значение х = {round(mot, 22)}', bg='black', fg='white')

tr1 = Label(window, text=f'Методом перебора получено значение х = {round(mob, 22)}', bg='black', fg='white')

if count != 0:

tr.configure(text=f'Методом итерации получено значение х = {mot}')

tr1.configure(text=f'Методом итерации получено значение х = {mob}')

tr.place(x=110, y=35)

tr1.place(x=110, y=55)

window.mainloop()

# 6.3 Результат / скриншот

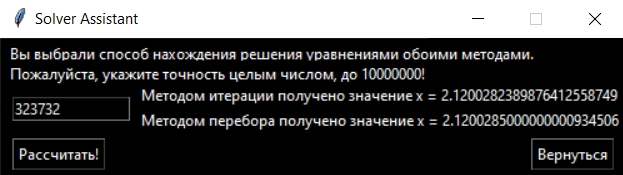


Рисунок 9 – результат работы

6.4 Промежуточный вывод: программа работает исправно как с методом итераций, так и с методом прямого перебора значений (параллельно) при заданных значениях пользователем (вплоть до 1E-23 и 1E+09 соответственно), выдавая конкретные ответы в окнах вывода.

# 7 Проектирование графического интерфейса

Задание 4

Для реализации графики использовался модуль Tkinter. Так как проект изначально строился на С++, но затем возникли трудности написания окон через WinAPI и последующих операций с ними. Было принято решение писать в среде с максимальной простотой, т.к. по заданию нужно лишь взаимодействие с пользователем. Поэтому был выбран модуль Tkinter т.к. данная библиотека содержит достаточное количество необходимых функций для связки основных вычислительных процедур с графическим интерфейсом.

Графический интерфейс в программе представлен несколькими функциями:  
1) comeback (Главное меню программы, начальное окно)

2) clicked (Обработка решения пользователя после выбора метода решения)

2) iterations (Окно с возможностью расчёта методом итераций (посл. прибл.))  
3) brute (Окно с возможностью расчёта методом полного перебора)  
4) special (Окно с возможностью для обоих подсчётов одновременно)

# 7.1 Comeback()

Как уже упоминалось ранее, эта функция в конечном итоге является самым первым окном, которое видит пользователь. В ней задаётся название окна “Solver assistant”, задний фон выполнен чёрным цветом, созданы виджеты – линии текста и кнопки (3 штуки, две типа Checkbutton, последняя типа Button). Пользователь может выбрать любой из методов для расчёта, и с помощью нажатия кнопки <Выбор сделан> получить желаемый способ. Сразу же напрашиваются два вопроса – что, если пользователь одновременно выберет оба способа? Или наоборот, просто забудет о выборе, и сразу нажмёт на кнопку с подтверждением выбора?

Эти своеобразные исключения необходимо обработать. За это отвечает функция clicked.

# 7.1.1 Код функции

def comeback():

global lbl, lbl1, lbl2, lb3, sel, sel1, bt, bt1, btn2

window.title("Solver Assistant")

window['bg'] = ['black']

lbl = Label(window, text="Задано иррациональное уравнение exp(-x) - x + 2 = 0", bg='black', fg='white')

lbl1 = Label(window, text="на интервале от -1.5 до 5.0 включительно.", bg='black', fg='white')

lbl2 = Label(window, text="Какой метод используем для вычисления переменной х?", bg='black', fg='white')

lb3 = Label(window, text='', bg='black', fg='red')

lbl.place(x=17, y=0)

lbl1.place(x=45, y=20)

lbl2.place(x=0, y=40)

sel = IntVar()

sel1 = IntVar()

bt = Checkbutton(window, text='Метод итераций (последовательных приближений)', onvalue=1, offvalue=0, variable=sel, bg='black', fg='white', selectcolor='red')

bt1 = Checkbutton(window, text='Метод прямого перебора чисел в рамках отрезка', onvalue=1, offvalue=0, variable=sel1, bg='black', fg='white', selectcolor='red')

bt.place(x=0, y=60)

bt1.place(x=0, y=80)

btn2 = Button(window, text="Выбор сделан", command=clicked, width='15', bg='black', fg='white')

btn2.place(x=5, y=110)

lb3.place(x=130, y=115)

window.resizable(0, 0

window.geometry("335x140")

window.mainloop()

# 7.1.2 Скриншот работы программы

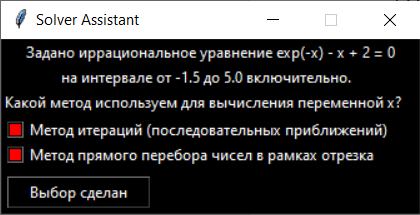


Рисунок 10 – меню выбора

# 7.2 Clicked()

Данная работа сразу запрашивает данные о выборе пользователя через переменные f, g, которые в свою очередь берут значения из переменных sel, sel1, которые в свою очередь напрямую привязаны к состояниям 2-х кнопок типа Checkbutton (Активное значение – 1, неактивное - 0). Если f, g оказались нулями, тогда функция лишь немного модернизирует начальное окно, красным текстом сообщив пользователю о необходимости выбора хотя бы одного варианта решения уравнения. Остальные выборы достаточно прямолинейны, так что если пользователь выберет метод итераций, то программа уйдёт в функцию iterations(), если выбран метод прямого перебора, то здесь будет иметь функция brute(), а если f, g обе равны единицам, тогда запускается особая функция special(), которая будет подсчитывать сразу два значения на основе введённой пользователем точности.

# 7.2.1 Код функции

f, g = sel.get(), sel1.get()

if f == 1 and g == 1:

special()

elif g == 1 and f == 0:

brute()

elif g == 0 and f == 1:

iterations()

else:

lb3.configure(text='Выберите хотя бы один вариант.')

# 7.2.2 Скриншот работы программы

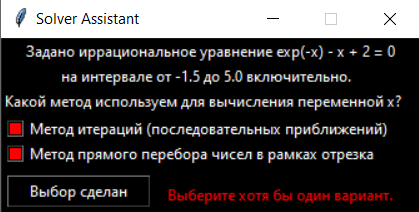


Рисунок 11 – пользователь не выбрал метод

# 7.3 Iterations()

Данная функция отвечает за подсчёт решения уравнения методом итераций (последовательных приближений). Она имеет в себе три подфункции – destroyer, callback, strange. Сперва меняется начальное окно по размеру, все прошлые виджеты исчезают с экрана, заменяясь новыми. Пользователь задаёт точность в окне типа Entry, которое получит на вход основная вычислительная процедура, а после этого выдаст ответ. Однако пользователь может ввести и неправильную точность, несмотря на предупреждение о границах введённой точности. Тогда функции callback и strange отвечают за получение значения путём нажатия клавиши Enter или простым нажатием кнопки <Рассчитать>, расположенной под полем ввода соответственно. Поэтому, если вдруг введено неверное значение, программа аналогично в первой функции красным текстом предупредит пользователя о необходимости соблюдения правил ввода.

# 7.3.1 Код функции

def iterations():

def destroyer():

new.destroy()

new1.destroy()

answer.destroy()

killjoy.destroy()

ass.destroy()

bi.destroy()

angriest.destroy()

if k:

wer.destroy()

comeback()

def callback(event):

global k, answer#ввод пользователя

if answer.get().isdigit():

if 0 < int(answer.get()) < 10000001:

k = True

else:

angriest.place(x=85, y=67)

else:

angriest.place(x=85, y=67)

def strange():

global k, answer

if answer.get().isdigit():#answer - поле для ввода, isdigit - вернёт false при проверке, если введено не число

if 0 < int(answer.get()) < 10000001:

k = True

else:

angriest.place(x=85, y=67)

else:

angriest.place(x=85, y=67)

bi = Button(window, text="Рассчитать!", bg='black', fg='white', command=strange)

bi.place(x=5, y=65)

global k, answer

window.geometry("310x110")

lbl.destroy()

lbl1.destroy()

lbl2.destroy()

bt.destroy()

bt1.destroy()

btn2.destroy()

window.update()

killjoy = Button(window, text='Вернуться', bg='black', fg='white', command=destroyer)

killjoy.place(x=240, y=65)

new = Label(window, text='Вы выбрали способ нахождения методом итераций.', bg='black', fg='white')

new1 = Label(window, text='Укажите точность натуральным числом до 10000000:', bg='black', fg='white')

ass = Label(window, text='(Для ввода нажмите Enter или кнопку под полем ввода)', bg='black', fg='white')

angriest = Label(window, text='Введите требуемое число!', bg='black', fg='red')

answer = Entry(window, bg='black', fg='white')#поле ввода

ass.place(x=0, y=90)

new.place(x=0, y=0)

new1.place(x=0, y=20)

answer.place(x=5, y=45)

window.update()

k = False#пользователь ничего не ввёл

while not k:

window.update()

window.bind("<Return>", callback)#после нажатия enter вызываем функцию callback

wer = Label(window, text=f'x = {method\_of\_iterations(int(answer.get()))}', bg='black', fg='white')#с помощью метода get получаем значение вводимое пользователем

new.configure(text=f'Для указанного значения точности {answer.get()} (итераций)')

new1.configure(text='Ответ к заданному уравнению: ')

new1.place(x=0, y=20)

wer.place(x=175, y=20)

window.geometry('310x95')

angriest.destroy()

ass.destroy()

# 7.3.2 Скриншоты работы программы

Работа окна в изначальном режиме:

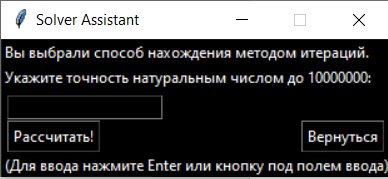


Рисунок 12 – меню метода итераций

Работа программы в случае неправильного ввода:

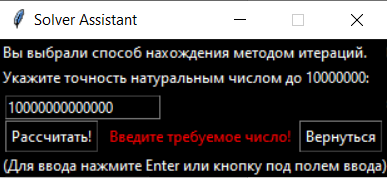


Рисунок 13 – пользователь ввёл неподходящее число

Работа программы в случае успешного ввода:

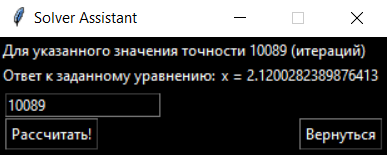


Рисунок 14 – результат работы

# 7.4 Brute()

Данная функция отвечает за подсчёт решения уравнения методом прямого перебора значений. Также имеет в себе 3 подфункции: destroyer, callback и strange. После того, как пользователь выбрал этот метод, окно остаётся таким же, но виджеты текста, ввода и кнопок заменяются. Пользователь также должен ввести значение точности и нажать кнопку <Рассчитать> или Enter на клавиатуре. Здесь же точность измеряется в другом размере. Если пользователь введёт недопустимое значение точности - strange и callback получат это значение и выведут предупреждающую табличку.

# 7.4.1 Код функции

def brute():

def destroyer():

new.destroy()

new1.destroy()

new2.destroy()

answer1.destroy()

bi.destroy()

limelighting.destroy()

if p:

wer.destroy()

comeback()

def callback(event):

global answer1, p

if answer1.get()[2:].isdigit() and answer1.get()[:2] == "0.":

if 1 > float(answer1.get()) > 0:

p = True

else:

new2.configure(text='Если разобрались со вводом,\n введите число согласно требованиям!', fg='red')

else:

new2.configure(text='Если разобрались со вводом,\n введите число согласно требованиям!', fg='red')

def strange():

global answer1, p

if answer1.get()[2:].isdigit() and answer1.get()[:2] == "0.":#срезы

if 1 > float(answer1.get()) > 0:

p = True

else:

new2.configure(text='Если разобрались со вводом,\n введите число согласно требованиям!', fg='red')

else:

new2.configure(text='Если разобрались со вводом,\n введите число согласно требованиям!', fg='red')

global answer1, p

bi = Button(window, text="Рассчитать!", bg='black', fg='white', command=strange)

bi.place(x=5, y=70)

window.geometry("450x100")

lbl.destroy()

lbl1.destroy()

lbl2.destroy()

bt.destroy()

bt1.destroy()

btn2.destroy()

window.update()

limelighting = Button(window, text="Вернуться", bg='black', fg='white', command=destroyer)

limelighting.place(x=380, y=70)

new = Label(window, text='Вы выбрали способ нахождения методом прямого перебора значений.', bg='black', fg='white')

new1 = Label(window, text='Укажите точность в размере от от 0.0000000000000000000001 до 1:', bg='black', fg='white')

new2 = Label(window, text='(После ввода нажмите клавишу Enter)\n(Или кнопку под полем ввода)',

bg='black', fg='white')

new2.place(x=150, y=45)

answer1 = Entry(window, bg='black', fg='white')

new.place(x=0, y=0)

new1.place(x=0, y=20)

answer1.place(x=5, y=50)

window.update()

p = False

while not p:

window.update()

window.bind("<Return>", callback)

wer = Label(window, text=f'x = {method\_of\_bruteforce(float(answer1.get()))}', bg='black', fg='white')

new.configure(text=f'Для указанного значения точности {answer1.get()} (ухода к нулю)')

new1.configure(text='Ответ к заданному уравнению составляет')

new2.destroy()

new1.place(x=0)

wer.place(x=240, y=20)

# 7.4.2 Скриншоты работы программы

Работа окна в изначальном режиме:

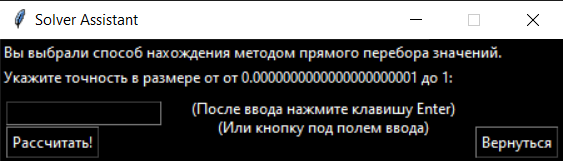


Рисунок 15 – меню метода прямого перебора

Работа программы в случае неправильного ввода:

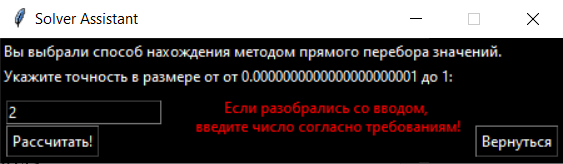


Рисунок 16 – пользователь ввёл неподходящее число

Работа программы в случае успешного ввода:

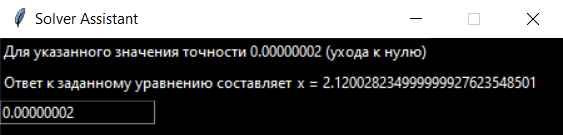


Рисунок 17 – результат работы

# 7.5 Special()

Данная функция отвечает за подсчёт решения уравнения двумя методами: методом итераций и методом прямого перебора значений (параллельно).

Функция special имеет в себе 3 подфункции: destroyer, callback и strange. Переходя к этому методу окно программы остаётся тем же (кроме как размера самого окна), но виджеты изменяются. Пользователь задаёт значение точности в окне Entry, которое получит программа и выдаст ответ. Также пользователь может задать значение недопустимое для программы, тогда после нажатия пользователем кнопки Enter с клавиатуры или кнопки <Рассчитать> в программе - callback и strange получают и обрабатывают данное значение и на экран выводится предупреждающая табличка.

# 7.5.1 Код функции

Код графической процедуры вставлен в пункте 6.2.

# 7.5.2 Скриншоты работы программы

Работа окна в изначальном режиме:

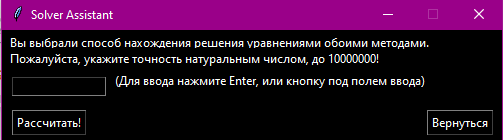


Рисунок 18 – меню обоих методов

Работа программы в случае неправильного ввода:

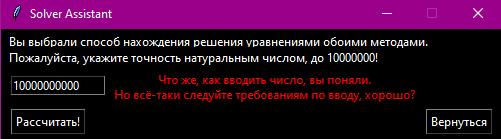


Рисунок 19 – пользователь ввёл неподходящее число

Работа программы в случае успешного ввода:

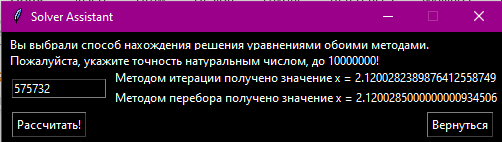


Рисунок 20 - результат работы

# 8 Оценка времени решения

Задание 5

Определить время вычисления по процедурам (2) и (3), сравнить и пояснить их. Определяем через модуль time в языке Python. Для увеличения точности расчёта времени, каждое измерение было рассчитано из арифметической суммы измерений (примерно десять тысяч слагаемых) при одинаковой, заданной пользователем точности вычисления корней исходного уравнения. Результат не округлён для увеличения количества чисел, после запятой.

# 8.1 Замер времени выполнения программы оптимизированного прямого перебора.

1: 0.000238200000000000911004605086418450810015201568603515625

0.1: 0.00023489999999999622293245238324743695557117462158203125

0.01: 0.0002787999999999957179142029417562298476696014404296875

0.001: 0.00029400000000000259259280710466555319726467132568359375

0.0001: 0.00026890000000000247037945655392832122743129730224609375

1e-05: 0.000281999999999997530419904023801791481673717498779296875

1e-06: 0.000323200000000002651834307698663906194269657135009765625

1e-07: 0.00036869999999999958362195684458129107952117919921875

1e-08: 0.000396000000000000518252107895023073069751262664794921875

1e-09: 0.000683299999999997798028061879449523985385894775390625

1e-10: 0.000524900000000001754063561065777321346104145050048828125

1e-11: 0.000406200000000002392486209146227338351309299468994140625

1e-12: 0.0003369000000000010874856570808333344757556915283203125

1e-13: 0.000335099999999997899546855251173838041722774505615234375

1e-14: 0.00035090000000000121094245741915074177086353302001953125

1e-15: 0.000342000000000002024602707706435467116534709930419921875

1e-16: 0.00039329999999999920579085710414801724255084991455078125

1e-17: 0.000480500000000001759037360216098022647202014923095703125

1e-18: 0.001959400000000000030553337637684307992458343505859375

1e-19: 0.01571779999999999011262019621426588855683803558349609375

1e-20: 0.1399972000000000160735424969971063546836376190185546875

1e-21: 1.343961900000000042609826778061687946319580078125

1e-22: 10.057450700000000409772837883792817592620849609375

1e-23: 121.35734750000000303771230392158031463623046875

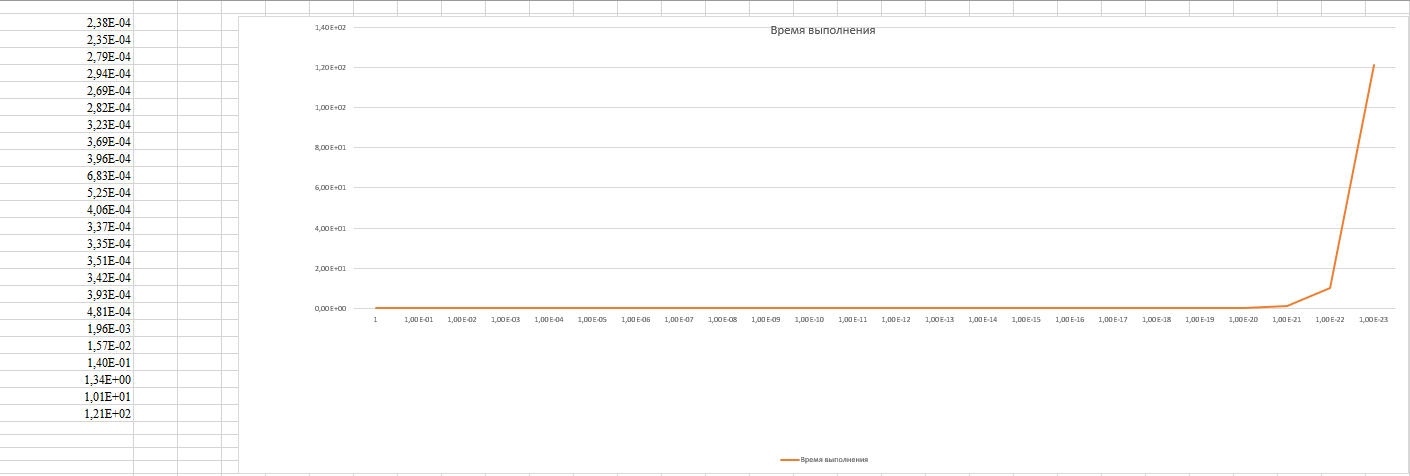


Рисунок 20 – график зависимости при методе перебора

Вывод: построив график, можно увидеть зависимость точности от времени исполнения: чем выше точность – тем выше время измерения.

# 8.2 Замер времени выполнения программы методом итераций.

1: 0.000027100000000002122391151715419255197048187255859375

10: 0.000010299999999997810906648965101339854300022125244140625

100: 0.00003419999999999812079209959847503341734409332275390625

1000: 0.00028129999999999821813645439760875888168811798095703125

10000: 0.001987999999999996714183936319386702962219715118408203125

100000: 0.03186250000000000193178806284777238033711910247802734375

1000000: 0.18078030000000000487858642372884787619113922119140625

10000000: 1.41797869999999992529637893312610685825347900390625

100000000: 14.0854815000000002100932761095464229583740234375

1000000000: 147.035737699999998540079104714095592498779296875

Chart, line chart

Description automatically generated

Рисунок 21 – график зависимости при методе итераций

Вывод: построив график, можно увидеть зависимость точности от времени исполнения: чем выше точность – тем выше время измерения

8.3 Пояснения.

Из приведенных выше результатов времени выполнения программы можно заметить эффективность оптимизированного метода прямого перебора чисел в сравнении с методом итераций. Почему же разница в возможности точности такая огромная? Всё дело в сложности алгоритмов вычислений.

Попробуем оценить метод итераций.

Видно, что сколько пользователь введет значение точности n, столько итераций программа и сделает. Следовательно, его сложность - O(n).

Оценим метод прямого перебора чисел.

Заметим, что программа сначала находит отрезок, в котором располагается ответ, в целых числах, дальше перебирает уже этот отрезок на уровне десятых, далее получившийся отрезок на уровне сотых, и так до вводимого пользователя значением точности n, следовательно, алгоритм выполняет по 10 операций за каждый знак после запятой. С учётом того, что тут точность представляется в виде числа с плавающей точкой, сложность алгоритма будет O(10\*lg(1/n) + 1). Но почему алгоритм зависает при значениях точности 1e(-10000000). Здесь уже идёт проблема в типе данных float, который позволяет нам хранить число с точностью 16 знаков после запятой. Значения искажаются, и отрезок получается уже не в 10 итераций - но случайно больше. Именно поэтому алгоритм зависает.

Идея для финальной оптимизации проста: можно было представить ответ в виде строки, к которой постоянно добавляется нужный символ 0-9, однако нельзя найти способа вставить такое число в вычисление значения функции, поскольку при приведении строки в тип Decimal теряется скорость, а приводя к float - теряется точность вычисления. Таким образом, заметно абсолютное превосходство метода оптимизированного перебора по сравнению с методом итераций за счёт способа реализации программы.

# 9 Вывод:

В ходе выполнения работы были изучены: метод итераций, метод прямого перебора значений.

Было проделано: найдено решение уравнения на указанном диапазоне с заданной точностью, использовав модуль «поиск решения» (Excel).

Найдено решение уравнения на указанном диапазоне с заданной точностью, используя метод прямого перебора значений в пределах диапазона.

Найдено решение уравнения на указанном диапазоне с заданной точностью, используя численный метод – Метод итераций (последовательных приближений).

Программа написана с применением элементов графического интерфейса пользователя (окна, кнопки, поля ввода, строк текста и т.д.)

Было измерено время работы каждой из основных вычислительных процедур с помощью многократных измерений, построены графики зависимости времени работы алгоритма от заданной точности.

Был оформлен комплект документации на проектную работу.