МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»**

# *ИНСТИТУТ ИКН*

*КАФЕДРА ИКТ*

#### *НАПРАВЛЕНИЕ 09.03.03*

#### *ДИСЦИПЛИНА Алгоритмы дискретной математики*

##### РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА

**на тему:** «Разработка и реализация программы добавление узла в нечеткое

бинарное дерево, используя метод центра тяжести и гармоническая

функция принадлежности». Вариант – 11.

*Студент группы БПИ-22-1 Зинченко Таисия Сергеевна*

*Руководитель* *работы д.т.н., проф. Халкечев Р.*

*Работа принята к защите*

*Работа сдана с оценкой*

Москва

2024

СОДЕРЖАНИЕ

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 3](#_Toc167918347)

[1.1 Определение нечеткого множества 3](#_Toc167918348)

[1.2 Асимметрическая гауссова функция 3](#_Toc167918349)

[1.3 Метод центра тяжести: 4](#_Toc167918350)

[1.4 Добавление элемента в узел. 4](#_Toc167918351)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 6](#_Toc167918352)

[2.1. Разработка программы 6](#_Toc167918353)

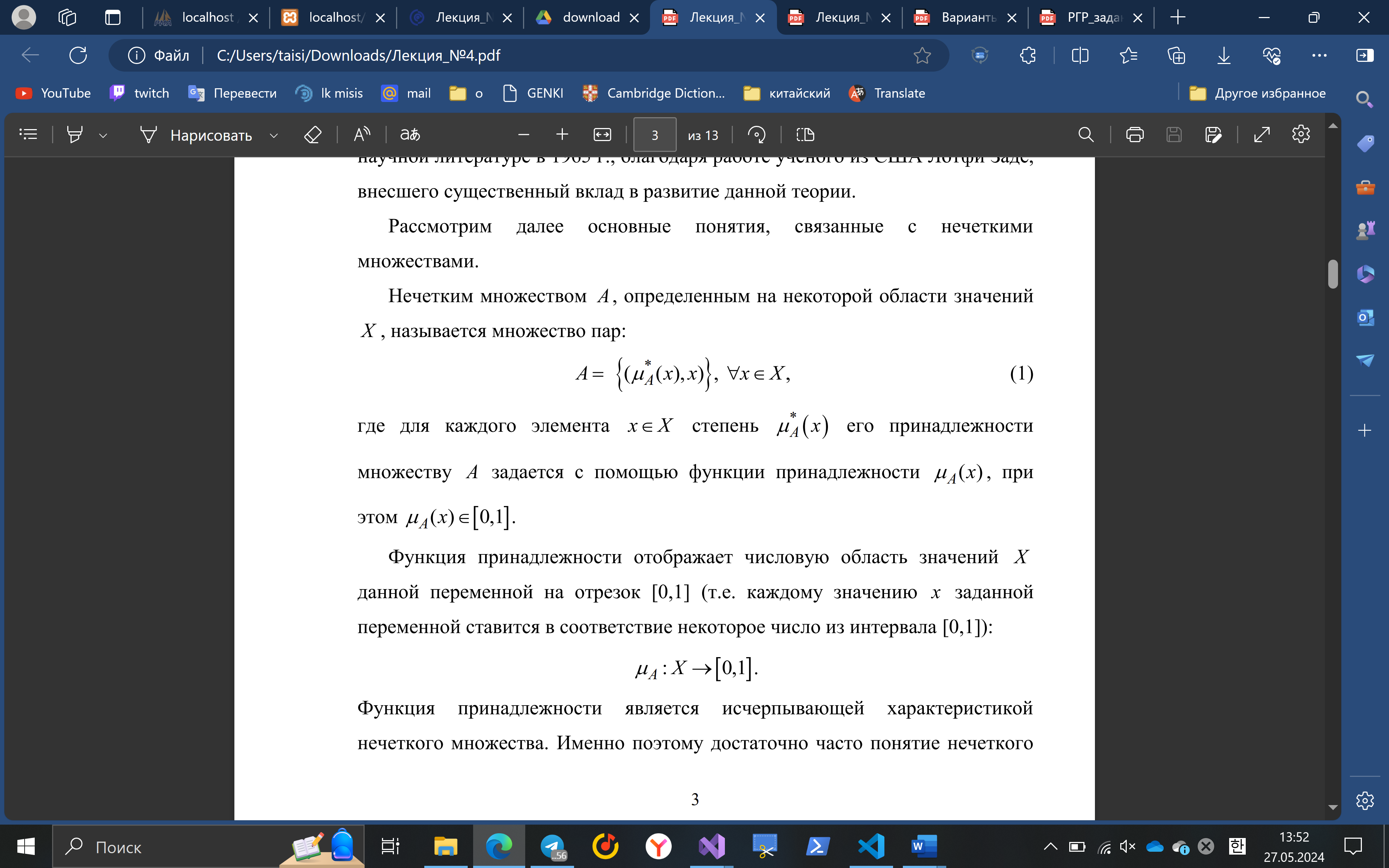
[2.2 Демонстрация работы программы 10](#_Toc167918354)

[3. ЛИСТИНГ ПРОГРАММЫ 12](#_Toc167918355)

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Определение нечеткого множества

Нечетким множеством А, определенным на некоторой области значений Х, называется множество пар:



1.2 Асимметрическая гауссова функция

Данная функция сочетает в себе преимущество гауссовой функции, связанное с бесконечной дифференцируемостью, с отсутствием недостатка, выражающегося в ее симметричности (см. рис. 1).

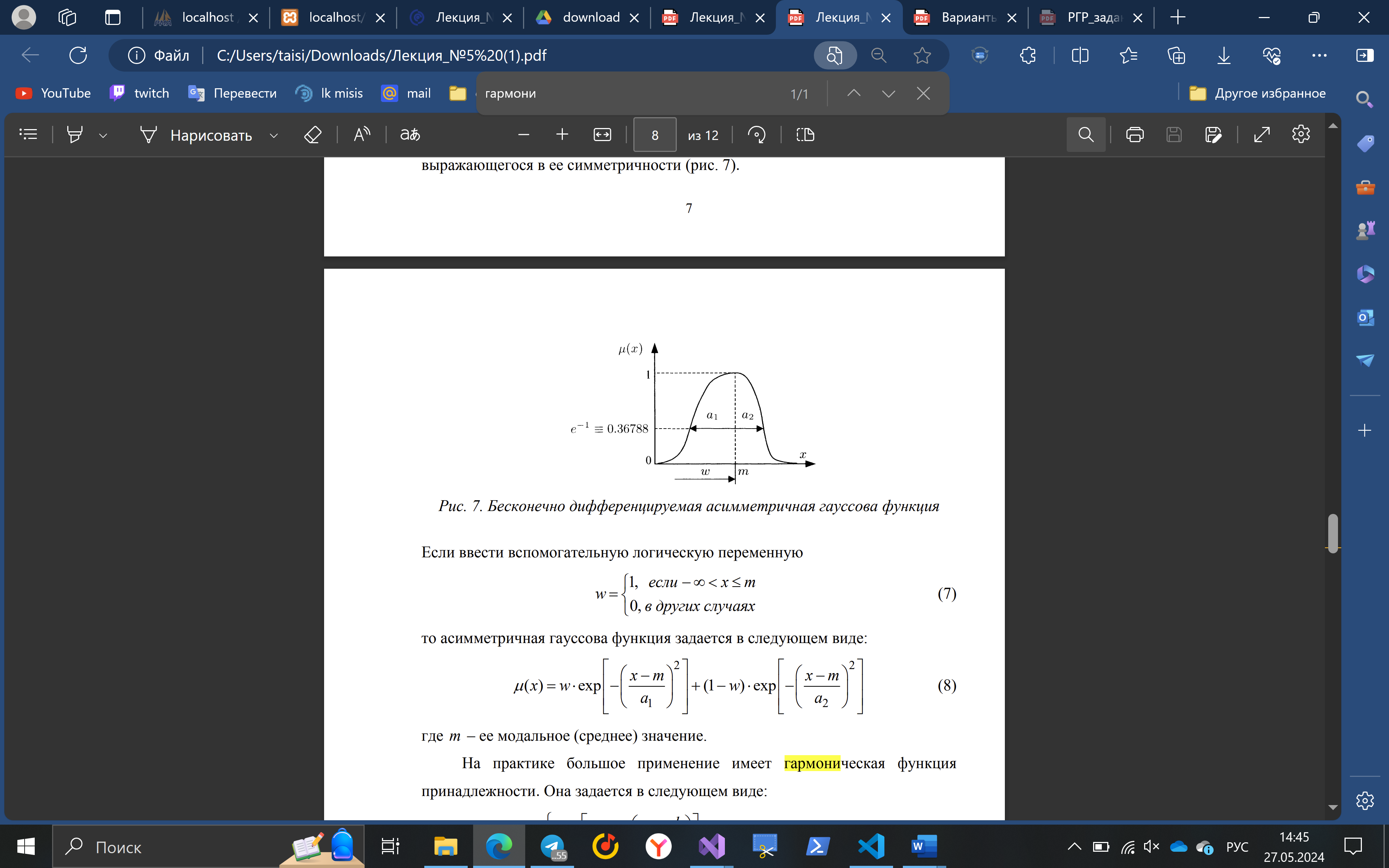
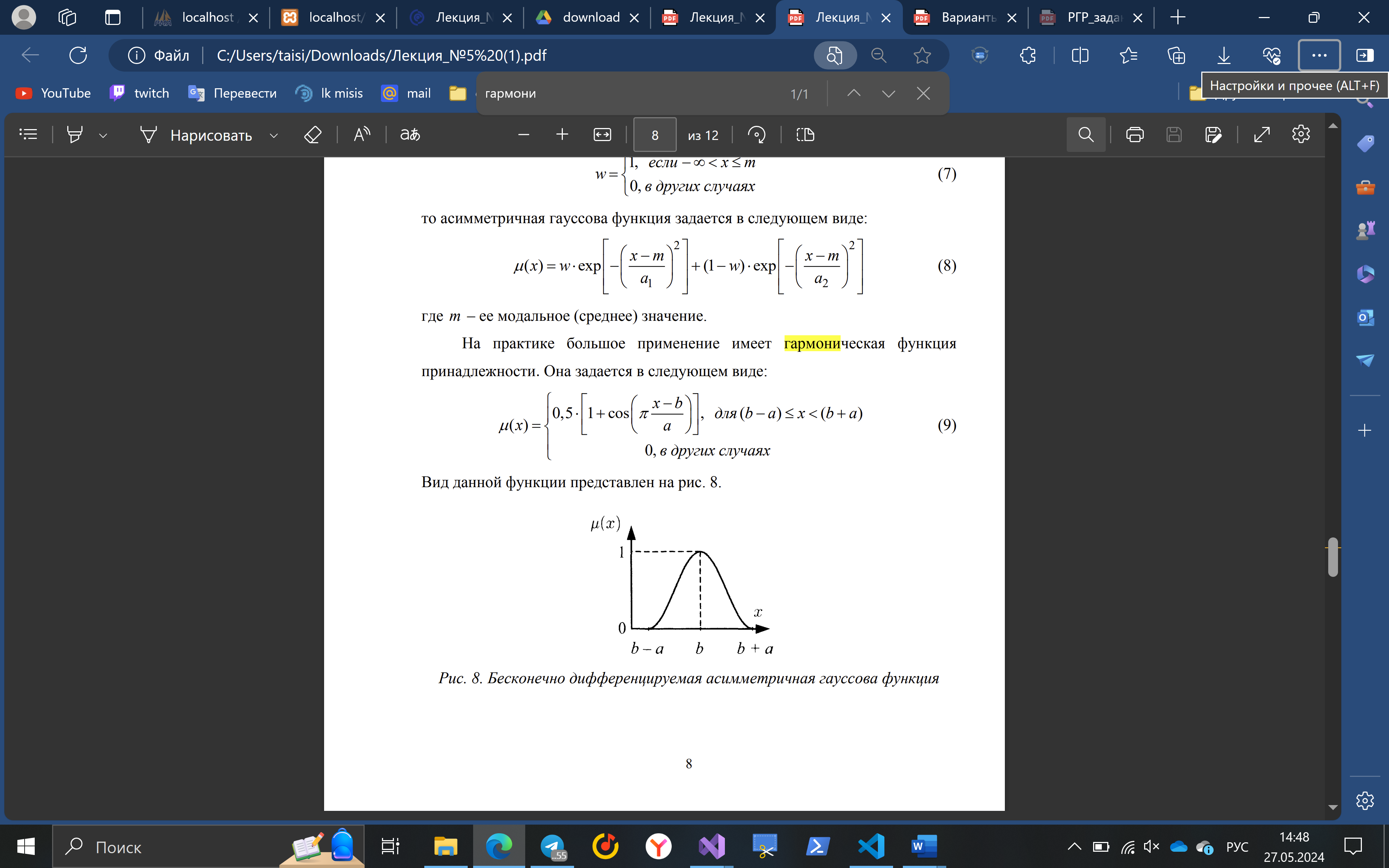


Рисунок 1 – Бесконечно дифференцируемая ассиметричная гауссова функция

На практике большое применение имеет гармоническая функция принадлежности. Она задается в следующем виде:



Вид данной функции представлен на рис. 2:

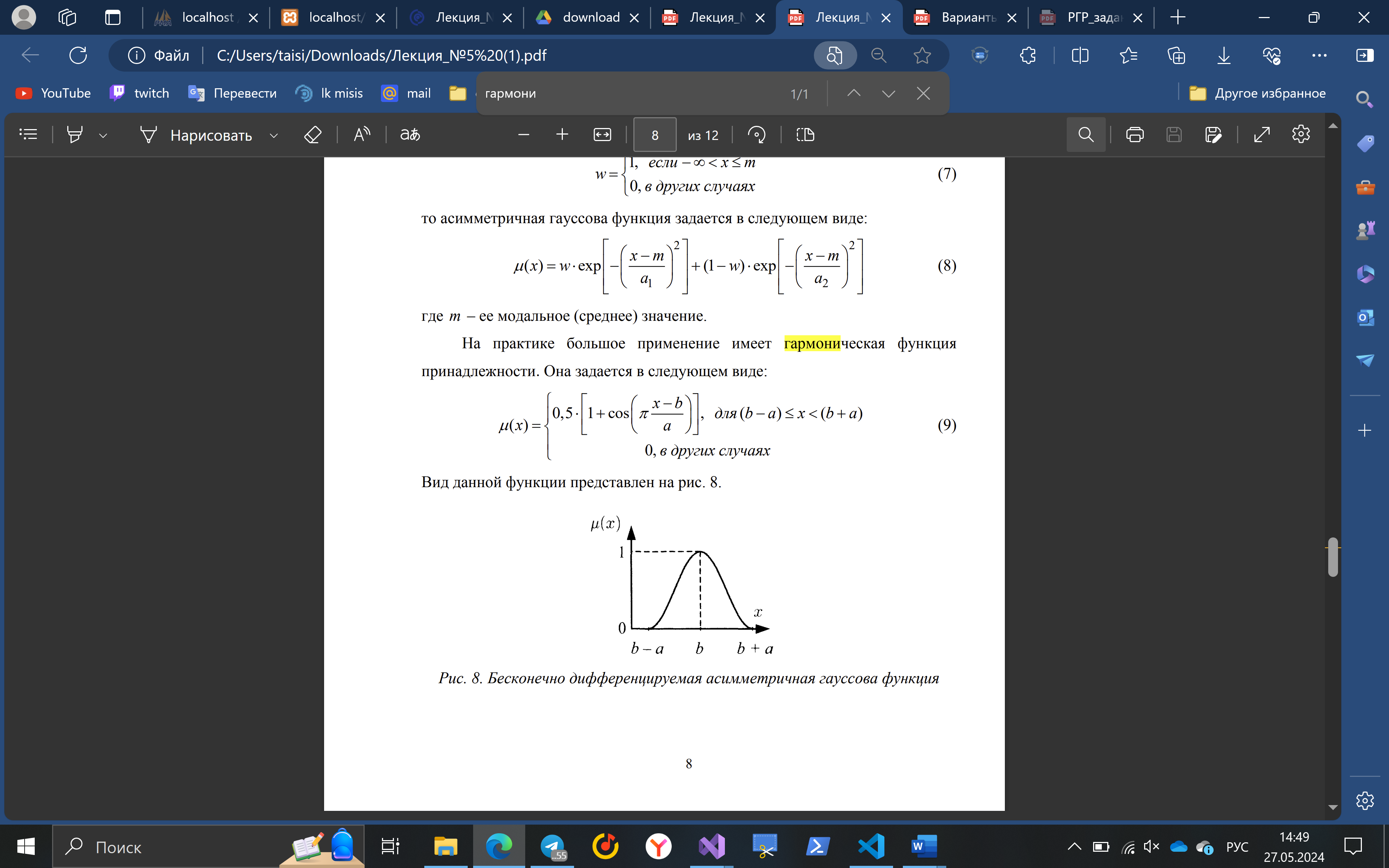
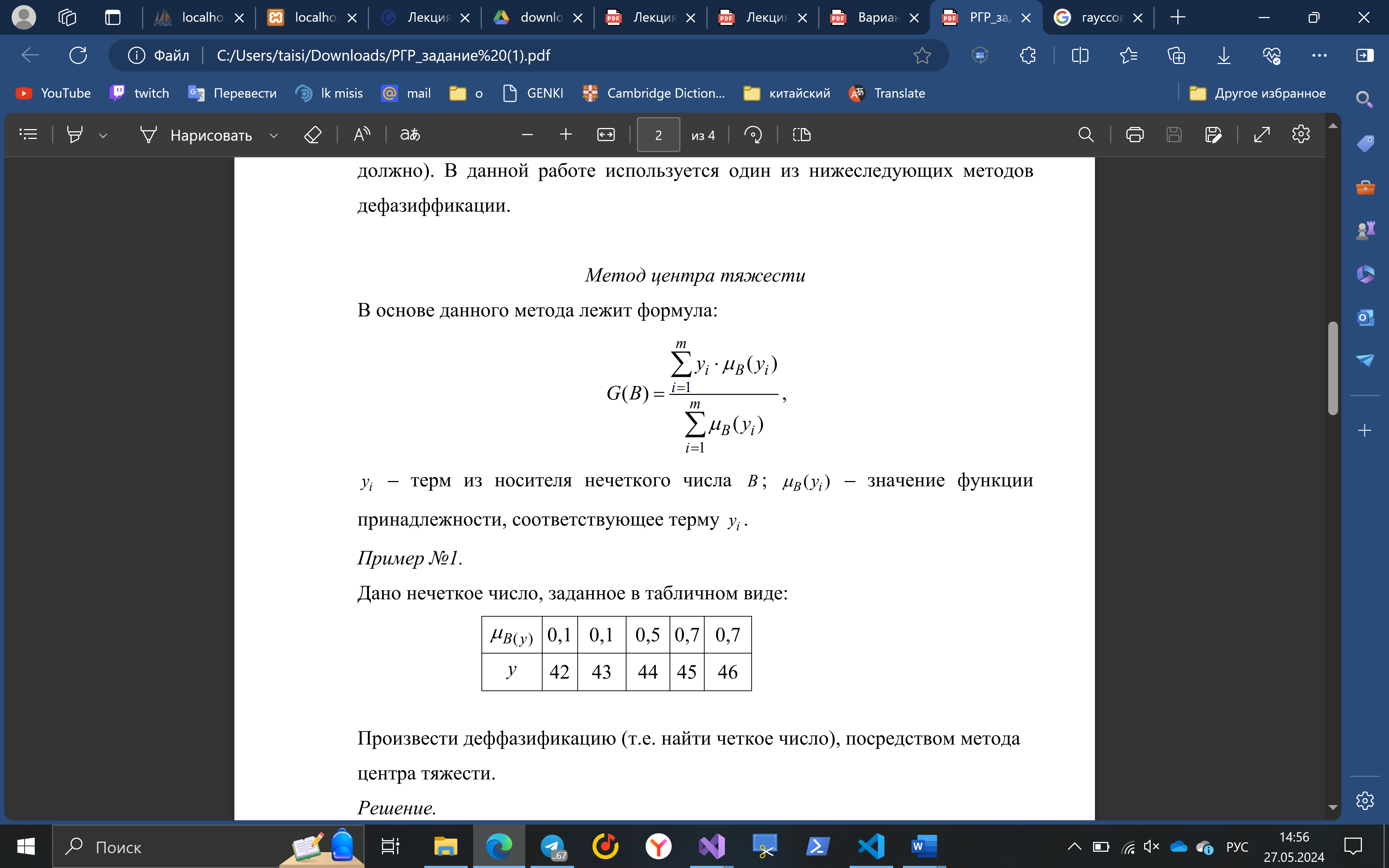


Рисунок 2 – Бесконечно дифференцируемая асимметричная гауссова функция

1.3 Метод центра тяжести:

*Дефаззификация* - это преобразование нечеткого множества в четкое число. Операция дефаззификации является необходимым элементом идентификации нелинейных зависимостей посредством настройки (обучения) нечетких баз знаний.

Метод центра тяжести является разновидностью дефаззификации и определяется по формуле:

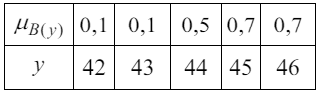


В данной формуле y­i – терм из носителя нечеткого числа B ; μ­B (yi) – значение функции принадлежности, соответствующее терму yi .

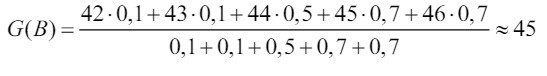
Для наглядного примера вычисления данного метода рассмотрим пример №1.

Пример №1.

Дано нечеткое число, заданное в табличном виде



Произвести деффазификацию (т. е. найти четкое число), посредством метода центра тяжести. Решение:



1.4 Добавление элемента в узел.

Алгоритм добавления узла в дерево:

1. Начнем с корневого узла дерева.

2. Если дерево пустое (корневой узел отсутствует), то новый узел становится корневым узлом.

3. Если дерево не пустое, начнем с корневого узла и будем спускаться по дереву для поиска места, куда можно добавить новый узел.

4. Сравниваем значение нового узла с текущим узлом:

- Если значение нового узла меньше значения текущего узла, двигаемся влево.

- Если значение нового узла больше значения текущего узла, двигаемся вправо.

- Продолжаем этот процесс до тех пор, пока не найдем подходящее место для нового узла.

5. Когда мы достигаем конечного узла (листового), добавляем новый узел либо в левую, либо в правую ветвь в зависимости от сравнения значений.

6. Устанавливаем связь между родительским узлом и новым узлом.

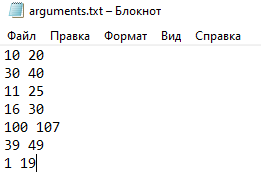
7. Новый узел успешно добавлен в дерево.

Это основной алгоритм добавления узла в бинарное дерево поиска. При реализации данного алгоритма важно учитывать особенности вашей структуры данных и языка программирования, который вы используете.

1. ПРАКТИЧЕСКАЯ ЧАСТЬ

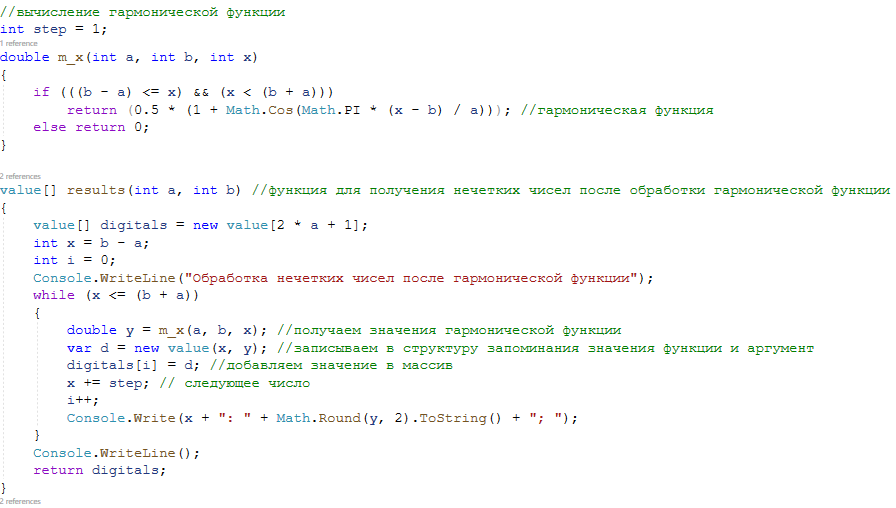
2.1. Разработка программы

Первый этап разработки – определение формата входных данных. В качестве записи входных значений использую текстовый файл (см. скрин. 1). Каждая строка файла – значения, представленные нечеткими числами a и b – границами промежутка.



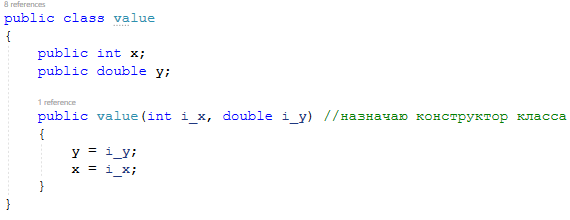
Скриншот 1 – Входные значения

После определения входных значений, приступаю к их обработке: присвоению шага изменения величины терма числа и использования гармонической функции принадлежности (см. скрин. 2).



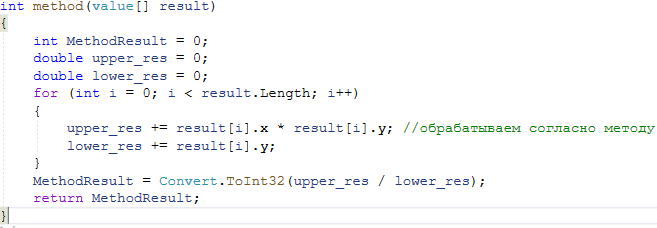
Скриншот 2 – Обработка входных значений

Для удобства проведения деффазификации чисел создаю массив класса value, в котором хранится аргумент и значение функции(см. скрин. 3).



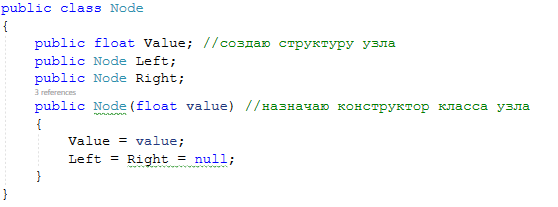
Скриншот 3 – Класс value

Далее обрабатываю полученные значения методом центра тяжести (см. скрин. 4).



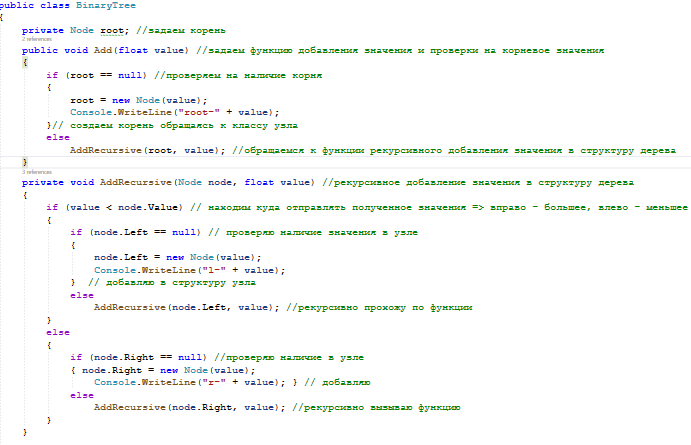
Скриншот 4 – Метод центра тяжести

Второй этап – разработка класса бинарного дерева. В данном этапе определяю класс узла Node (см. скрин. 5).

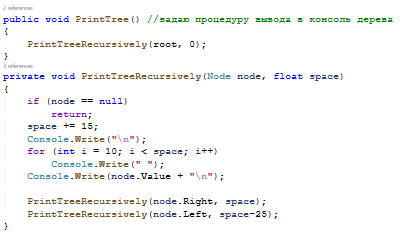


Скриншот 5 – Класс узла бинарного дерева

После разработки структуры узла перехожу к классу бинарного дерева BinaryTree (см. скрин. 6 и 7). В данной структуре используется рекурсивная проверка значений. Далее обозначаю процедуры вывода класса в консоли.



Скриншот 6 – Класс BinaryTree



Скриншот 7 – Класс BinaryTree

Перехожу к третьему этапу – добавление в дерево значений. Для этого использую функцию add\_new (см. скрин. 8). В функции определяю количество терм и обрабатываю значения через функцию и метод, получаю четкое число, добавляю его в бинарное дерево.



Скриншот 8 – Добавление узла

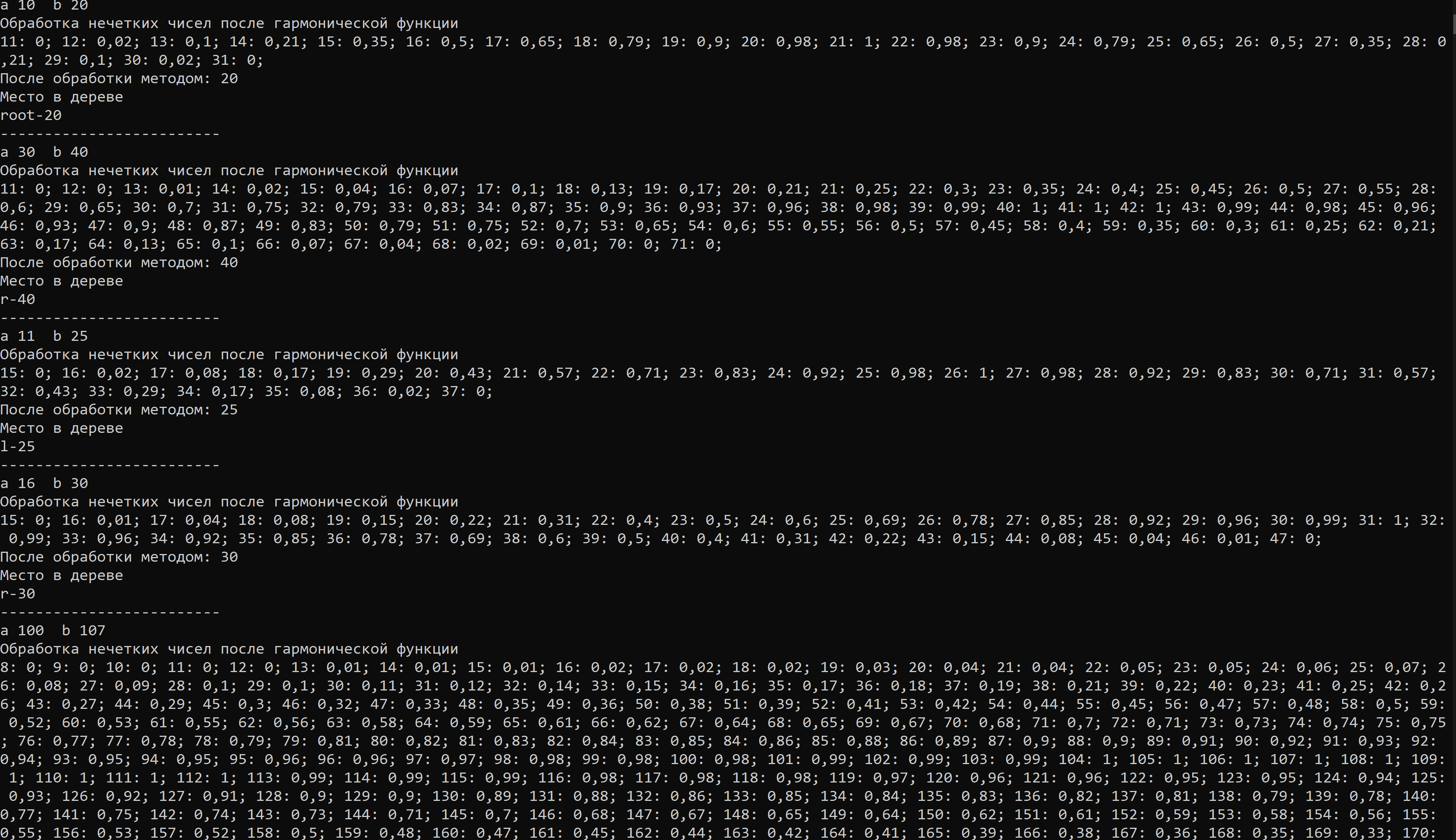
Перехожу к последнему этапу – соединению частей кода в одно целое (см. скрин. 9). Использую файловый обработчик и подготовленные функции, произвожу проверку данных и вывожу дерево.



Скриншот 9 – Соединение частей кода

2.2 Демонстрация работы программы

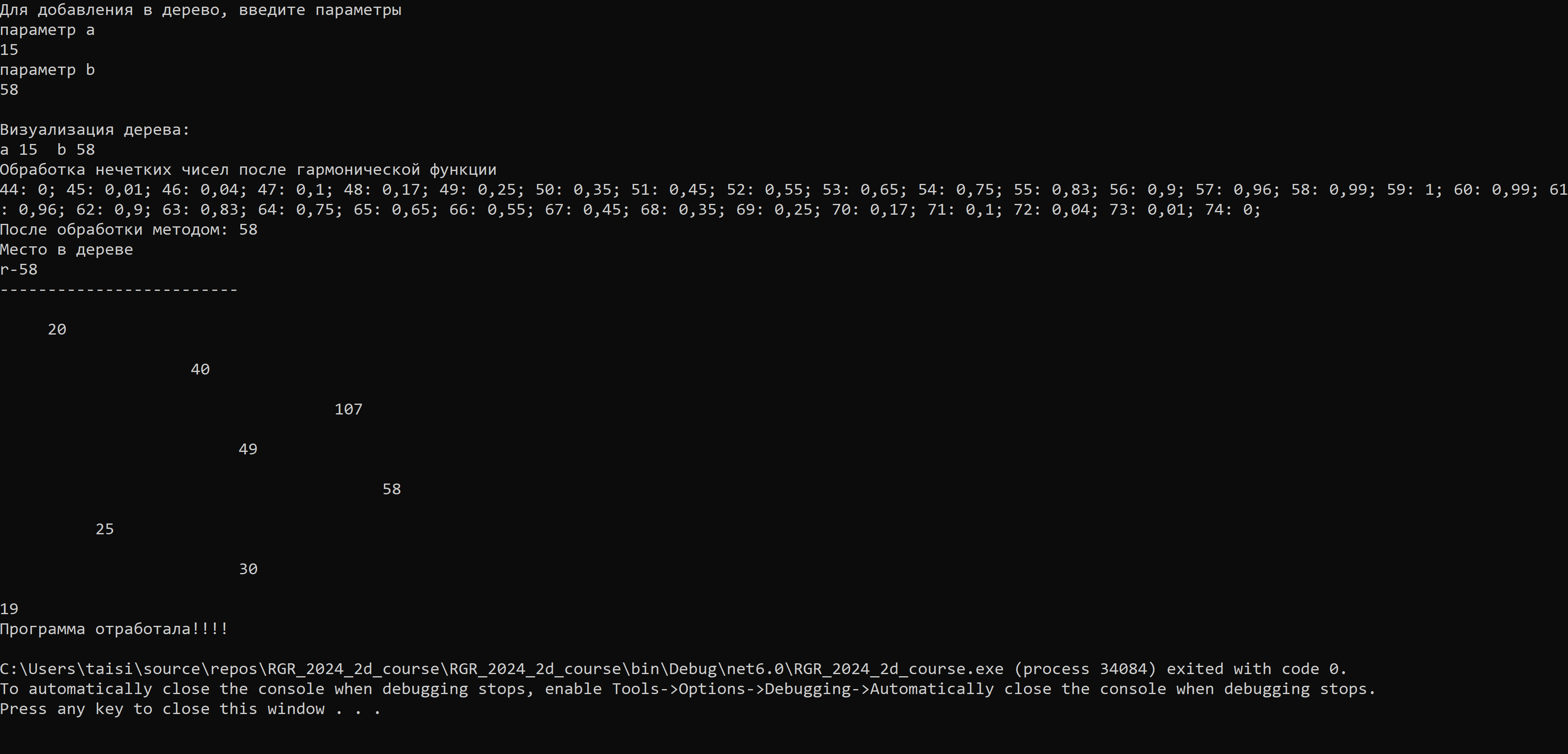
Результат работы программы представлен в консоли (см. скрин. 10, 11, 12).



Скриншот 10 – Демонстрация работы программы



Скриншот 11 – Демонстрация работы программы



Скриншот 12 – Демонстрация работы программы

1. ЛИСТИНГ ПРОГРАММЫ

1. Файл secretvalues.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RGR\_2024\_2d\_course

{

public class value

{

public int x;

public double y;

public value(int i\_x, double i\_y) //назначаю конструктор класса узла

{

y = i\_y;

x = i\_x;

}

}

}

2. Файл node.cs:

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RGR\_2024\_2d\_course

{

public class Node

{

public float Value; //создаю структуру узла

public Node Left;

public Node Right;

public Node(float value) //назначаю конструктор класса узла

{

Value = value;

Left = Right = null;

}

}

}

3. Файл binaryTree.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace RGR\_2024\_2d\_course

{

public class BinaryTree

{

private Node root; //задаем корень

public void Add(float value) //задаем функцию добавления значения и проверки на корневое значения

{

if (root == null) //проверяем на наличие корня

{

root = new Node(value);

Console.WriteLine("root-" + value);

}// создаем корень обращаясь к классу узла

else

AddRecursive(root, value); //обращаемся к функции рекурсивного добавления значения в структуру дерева

}

private void AddRecursive(Node node, float value) //рекурсивное добавление значения в структуру дерева

{

if (value < node.Value) // находим куда отправлять полученное значения => вправо - большее, влево - меньшее

{

if (node.Left == null) // проверяю наличие значения в узле

{

node.Left = new Node(value);

Console.WriteLine("l-" + value);

} // добавляю в структуру узла

else

AddRecursive(node.Left, value); //рекурсивно прохожу по функции

}

else

{

if (node.Right == null) //проверяю наличие в узле

{ node.Right = new Node(value);

Console.WriteLine("r-" + value); } // добавляю

else

AddRecursive(node.Right, value); //рекурсивно вызываю функцию

}

}

public void PrintTree() //задаю процедуру вывода в консоль дерева

{

PrintTreeRecursively(root, 0);

}

private void PrintTreeRecursively(Node node, float space)

{

if (node == null)

return;

space += 15;

Console.Write("\n");

for (int i = 10; i < space; i++)

Console.Write(" ");

Console.Write(node.Value + "\n");

PrintTreeRecursively(node.Right, space);

PrintTreeRecursively(node.Left, space-25);

}

}

}

4. Файл Program.cs:

using RGR\_2024\_2d\_course;

using System;

using System.ComponentModel.DataAnnotations;

//вычисление гармонической функции

int step = 1;

double m\_x(int a, int b, int x)

{

if (((b - a) <= x) && (x < (b + a)))

return (0.5 \* (1 + Math.Cos(Math.PI \* (x - b) / a))); //гармоническая функция

else return 0;

}

value[] results(int a, int b) //функция для получения нечетких чисел после обработки гармонической функции

{

value[] digitals = new value[2 \* a + 1];

int x = b - a;

int i = 0;

Console.WriteLine("Обработка нечетких чисел после гармонической функции");

while (x <= (b + a))

{

double y = m\_x(a, b, x); //получаем значения гармонической функции

var d = new value(x, y); //записываем в структуру запоминания значения функции и аргумент

digitals[i] = d; //добавляем значение в массив

x += step; // следующее число

i++;

Console.Write(x + ": " + Math.Round(y, 2).ToString() + "; ");

}

Console.WriteLine();

return digitals;

}

int method(value[] result)

{

int MethodResult = 0;

double upper\_res = 0;

double lower\_res = 0;

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

{

upper\_res += result[i].x \* result[i].y; //обрабатываем согласно методу

lower\_res += result[i].y;

}

MethodResult = Convert.ToInt32(upper\_res / lower\_res);

return MethodResult;

}

BinaryTree add\_new(int a, int b, BinaryTree tree)

{

Console.WriteLine("a " + a + " b " + b);

var res = new value[2 \* a + 1];

int val;

res = results(a, b);

val = method(res);

Console.WriteLine("После обработки методом: " + val);

Console.WriteLine("Место в дереве");

tree.Add(val);

Console.WriteLine("-------------------------");

return tree;

}

BinaryTree tree = new BinaryTree();

try

{

string filePath = "C:\\Users\\taisi\\source\\repos\\RGR\_2024\_2d\_course\\RGR\_2024\_2d\_course\\arguments.txt";

if (File.Exists(filePath))

{

//Pass the file path and file name to the StreamReader constructor

using (StreamReader sr = new StreamReader(filePath))

{

string line;

int c = 0;

while ((line = sr.ReadLine()) != null)

{

string[] subs = line.Split(' ');

int a = int.Parse(subs[0]);

int b = int.Parse(subs[1]);

tree = add\_new(a, b, tree);

c++;

}

if (c<7)

{

Console.WriteLine("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");

Console.WriteLine("Введите не менее семи элементов");

return;

}

}

Console.WriteLine("\nВизуализация дерева:");

tree.PrintTree();

}

Console.WriteLine("Для добавления в дерево, введите параметры");

Console.WriteLine("параметр a");

int an = int.Parse(Console.ReadLine());

Console.WriteLine("параметр b");

int bn = int.Parse(Console.ReadLine());

if ((an <= bn)&&((an)>0))

{

Console.WriteLine("\nВизуализация дерева:");

add\_new(an, bn, tree).PrintTree();

}

else { Console.WriteLine("Допускаются неотрицательные числа, где a<b"); }

}

catch (Exception e)

{

Console.WriteLine("Ошибка: " + e.Message);

}

finally

{

Console.WriteLine("Программа отработала!!!!");

}