**Санкт-Петербургский Национальный Исследовательский Университет ИТМО**

**Лабораторная работа №1**

по дисциплине

*«Вычислительная математика»*

Выполнил: Анисимов М. Д.

Группа: Р3233

Преподаватель: Перл О. В.

Санкт-Петербург

2024 г.

**Цель работы**

Разработать программу, которая вычисляет интерполяционный полином Лагранжа. В качестве аргументов программа принимает строку со списком значений аргумента для узлов интерполяции, строку со списком значений аргумента для узлов интерполяции и строку со значением аргумента, для которого находится значение интерполяционного полинома

**Описание метода**

Метод основан на построении интерполяционного полинома Лагранжа

и нахождении его значения в заданной точке x. В формле полинома Лагранжа - список значений функции для соответствующего значения аргумента для узлов интерполяции, – значения аргумента для узла интерполяции, – значение аргумента, для которого мы ищем интерполяционный полином.

Результат вычисления метода представляет собой вещественное число, являющиеся значением интерполяционного полинома в точке х

**Полный код программы**

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

import static java.util.stream.Collectors.toList;

class Result {

public static double interpolate\_by\_lagrange(List<Double> x\_axis, List<Double> y\_axis, double x) {

double total\_result = 0.0;

double nominator = 1.0;

double denominator = 1.0;

double current\_argument = 0.0;

int x\_axis\_size = x\_axis.size();

for(int i=0; i<x\_axis\_size; i++){

current\_argument = x\_axis.get(i);

for(int j=0; j<x\_axis\_size; j++){

if(i != j) {

nominator = nominator \* (x - x\_axis.get(j));

denominator = denominator \* (current\_argument - x\_axis.get(j));

if(denominator ==0) {

nominator = 0;

denominator = 1;

break;

}

}

}

total\_result = total\_result + (nominator / denominator) \* y\_axis.get(i);

nominator = 1.0;

denominator = 1.0;

}

return total\_result;

}

}

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(System.out));

int axisCount = Integer.parseInt(bufferedReader.readLine().trim());

List<Double> x\_axis = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))

.map(Double::parseDouble)

.collect(toList());

List<Double> y\_axis = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))

.map(Double::parseDouble)

.collect(toList());

double x = Double.parseDouble(bufferedReader.readLine().trim());

double result = Result.interpolate\_by\_lagrange(x\_axis, y\_axis, x);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

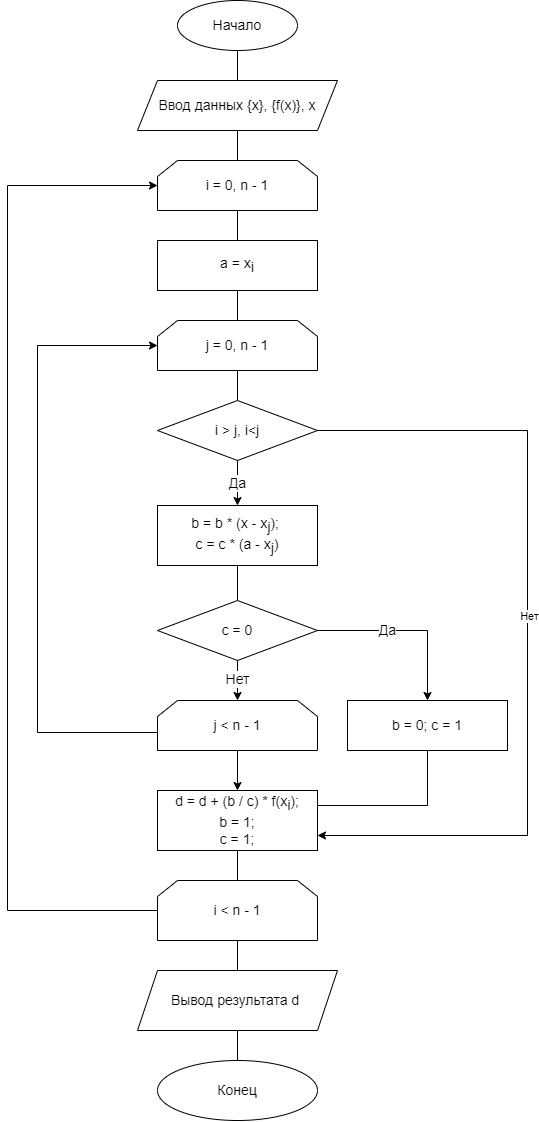
}

**Примеры работы программы**

(Зелёные цифры – входные данные. Белые цифры – результат)

|  |  |
| --- | --- |
| Пример №1 |  |
| Пример №2 |  |
| Пример №3 |  |
| Пример №4 |  |
| Пример №5 |  |

**Блок-схема работы программы**



**Выводы по работе программы**

Данный метод весьма эффективен для интерполяции данных, так как для вычислений используется несложная формула, благодаря которой мы можем находить интерполяционный полином и его значение. Однако его главным недостатком является слишком медленная скорость вычисления при большом наборе данных, так необходимо вычислять большое количество базисных полиномов. В этом случае эффективнее использовать полином Ньютона. Алгоритмическая сложность вычисления полинома Лагранжа в моей программ составляет O(n2) (В худшем случае), что является не самой лучшей скоростью вычисления.