

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

**Интерполирование функции многочленом Ньютона и многочленом  
Лагранжа**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к лабораторной работе №3

по дисциплине

**Вычислительная Математика**

РУКОВОДИТЕЛЬ:

Суркова Анна Сергеевна

\_\_\_\_\_  
(подпись)

СТУДЕНТ:

Цветков Николай Максимович

\_\_\_\_\_  
(подпись)

19-ИВТ-3

Работа защищена «\_\_» \_\_\_\_\_

С оценкой \_\_\_\_\_

Нижний Новгород 2021

## Оглавление

Цель.....	3
Постановка задачи .....	4
Теоретические сведения .....	6
Многочлен Ньютона.....	6
Многочлен Лагранжа для неравноотстоящих узлов .....	9
Многочлен Лагранжа для равноотстоящих узлов .....	11
Расчетные данные.....	13
Листинг разработанной программы .....	15
Результаты работы программы .....	18
Вывод.....	19

### Цель

Закрепление знаний и умений по интерполированию функций с помощью многочленов Ньютона и Лагранжа

### Постановка задачи

1) Вычислить значение функции при данных значениях аргумента, оценить погрешность:

а) используя первую или вторую интерполяционную формулу Ньютона, в зависимости от значения аргумента;

б) с помощью интерполяционного многочлена Лагранжа, используя формулу для равноотстоящих узлов.

x	y	№ варианта	Значение аргумента				
			$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
0,01	0,991824	4	0,11	0,13	0,62	0,005	0,44
0,06	0,951935						
0,11	0,91365						
0,16	0,876905						
0,21	0,841638						
0,26	0,807789						
0,31	0,775301						
0,36	0,74412						
0,41	0,714193						
0,46	0,68547						
0,51	0,657902						
0,56	0,631442						
0,61	0,611242						

Найти приближенное значение функции при данных значениях аргумента с помощью интерполяционного многочлена Лагранжа, если функция задана в неравноотстоящих узлах таблицы, оценить погрешность

x	y	№ варианта	$x_1$	$x_2$
0.41	2.57418	4	0.616	0.444
0.46	2.32513			0.555

0.52	2.09336			
0.60	1.86208			
0.65	1.74926			
0.72	1.62098			

## Теоретические сведения

### Многочлен Ньютона

Служит для построения многочлена  $n$ -й степени, который совпадает в  $(n+1)$  точке со значениями неизвестной искомой функции  $y = f(x)$ .

Пусть в точках  $x_0, x_1, \dots, x_{n+1}$  значения функции  $y = f(x)$  равны соответственно  $y_0 = f(x_0), y_1 = f(x_1), \dots, y_{n+1} = f(x_{n+1})$ .

Построим интерполяционный многочлен Ньютона с помощью метода неопределенных коэффициентов. Для этого запишем искомый многочлен в виде

$$P_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + b_3(x - x_0)(x - x_1)(x - x_2) + \dots + b_n(x - x_0)\dots(x - x_{n-1}). \quad (1)$$

Последовательно подставляя в формулу (1) вместо  $x$  данные значения  $x_0, x_1, \dots, x_{n+1}$ , получим для нахождения неопределенных коэффициентов  $b_0, b_1, \dots, b_n$  «треугольную» систему уравнений

$$\begin{cases} y_0 = b_0, \\ y_1 = b_0 + b_1(x_1 - x_0), \\ y_2 = b_0 + b_1(x_2 - x_0) + b_2(x_2 - x_0)(x_2 - x_1), \\ \dots, \\ y_{n+1} = b_0 + b_1(x_{n+1} - x_0) + \dots + b_n(x_{n+1} - x_0)\dots(x_{n+1} - x_n) \end{cases}$$

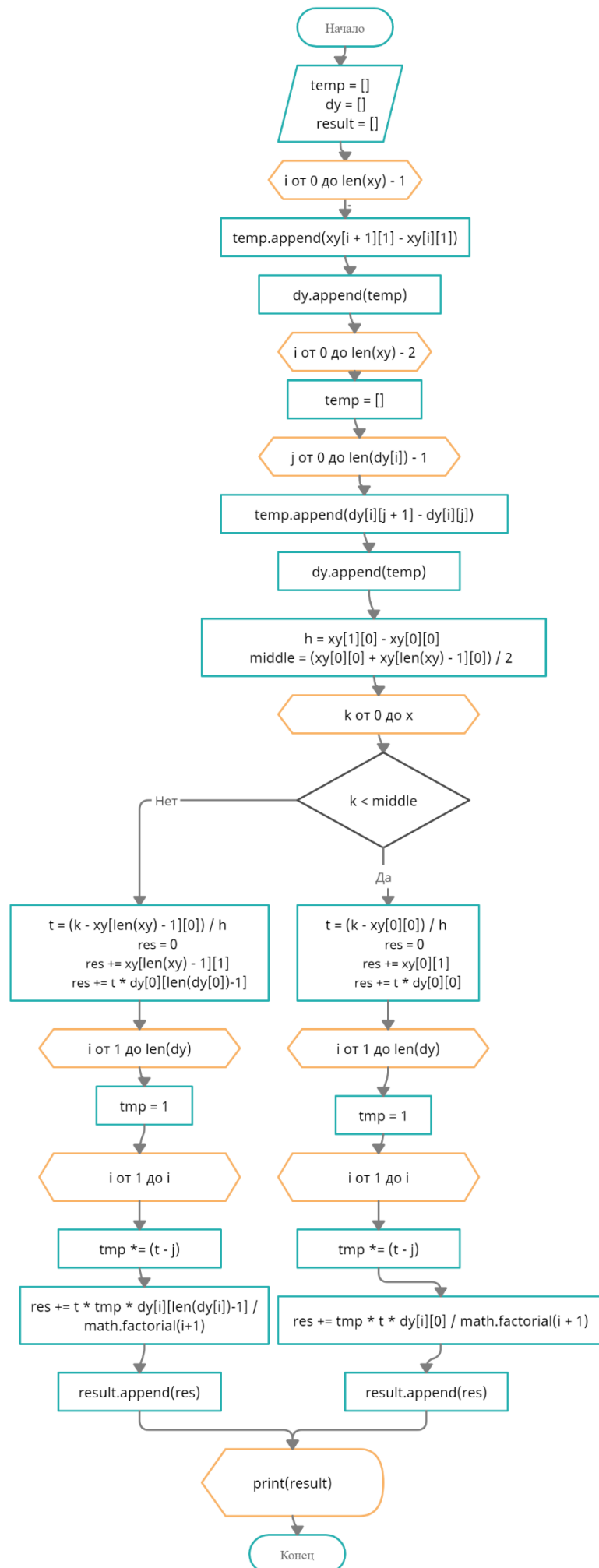
(при подстановке в равенство (1) вместо  $x$  числа  $x_0$  в правой части равенства обратились в нуль все слагаемые, кроме первого: там везде был множитель  $(x - x_0)$ , обратившийся в нуль; при подстановке  $x = x_1$  обратились в нуль все слагаемые, кроме первого и второго – они содержат множитель  $(x - x_1)$  и т.д.).

Полученную систему удобно решать: из первого её уравнения находим свободный член искомого многочлена  $b_0$ ; подставив его во второе уравнение, находим коэффициент  $b_1$  при первой степени  $x$  в искомом многочлене:

$$b_1 = \frac{y_1 - b_0}{x_1 - x_0}$$

и т.д.

Для интерполяционного многочлена Ньютона можно выписать явные выражения коэффициентов через данные задачи, а также и оценки точности замены неизвестной функции  $f(x)$  этим многочленом.





## Многочлен Лагранжа для неравноотстоящих узлов

Для решения задачи интерполяции функцию  $y = f(x)$  заменяют многочленом  $L_n(x)$  степени не выше  $n$ , который используют для приближенного вычисления значений функции. Многочлен полностью определяется требованием, чтобы его значения и значения  $f(x)$  совпадали в узлах интерполяции:

$$f(x_0) = L_n(x_0), f(x_1) = L_n(x_1), \dots, f(x_n) = L_n(x_n). \quad (1)$$

Будем искать многочлен в виде

$$L_n(x) = \sum_{k=0}^n c_k \prod_{\substack{j=0 \\ j \neq k}}^n (x - x_j), \quad (2)$$

где символ  $\prod$  называется знаком произведения, а выражение  $\prod_{j=0}^n a_j$  представляет собой сокращенную запись произведения  $a_0 \cdot \dots \cdot a_n$ . Например,

$$\begin{aligned} \sum_{k=0}^2 c_k \prod_{\substack{j=0 \\ j \neq k}}^2 (x - x_j) &= \\ &= c_0(x - x_1)(x - x_2) + c_1(x - x_0)(x - x_2) + c_2(x - x_0)(x - x_1). \end{aligned}$$

Для решения задачи интерполяции необходимо определить коэффициенты  $c_0, c_1, \dots, c_n$  многочлена (2).

Положим в формуле (2)  $x = x_i$  и используем условия (1). Получим

$$y_i = f(x_i) = L_n(x_i) = c_i \prod_{\substack{j=0 \\ j \neq i}}^n (x_i - x_j),$$

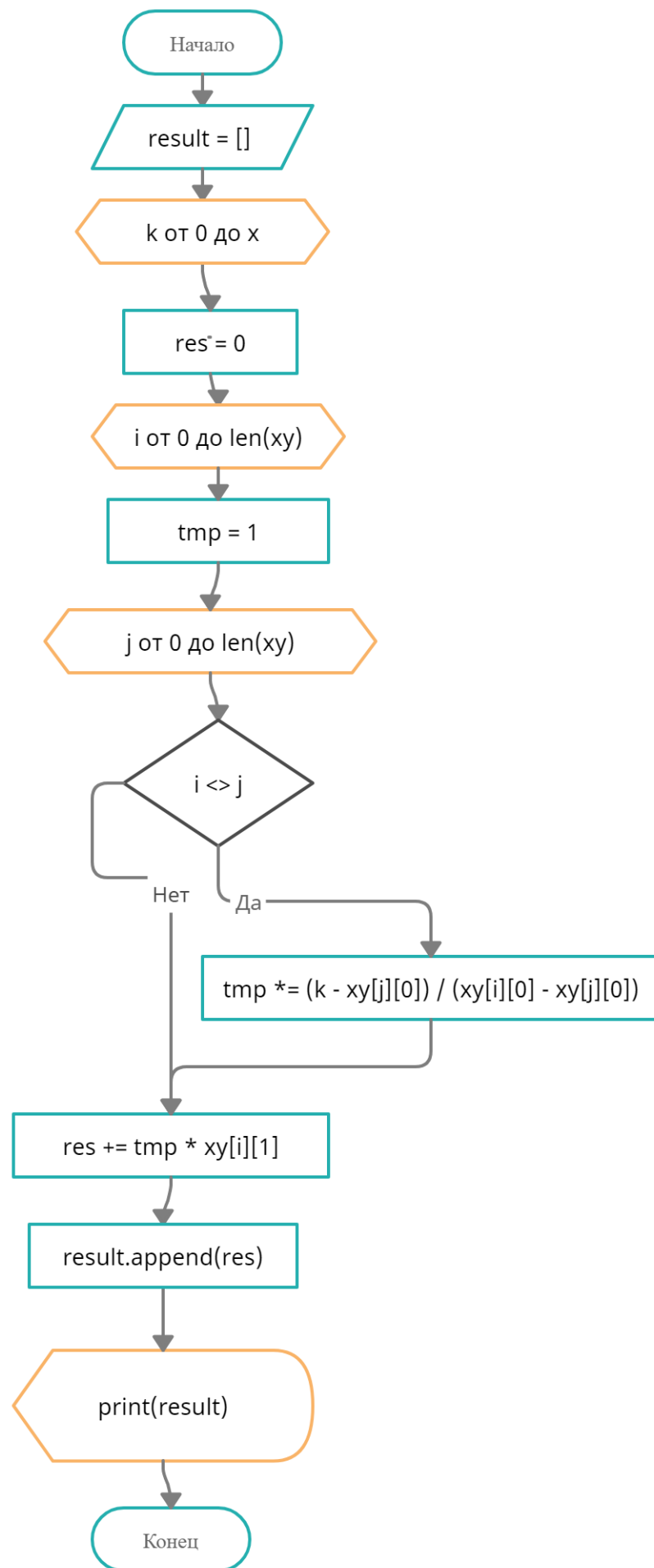
следовательно,

$$c_i = y_i / \prod_{\substack{j=0 \\ j \neq i}}^n (x_i - x_j).$$

Таким образом, искомый многочлен имеет вид

$$L_n(x) = \sum_{k=0}^n y_k \prod_{\substack{j=0 \\ j \neq k}}^n \frac{x - x_j}{x_k - x_j}. \quad (3)$$

Он называется **интерполяционным многочленом Лагранжа для неравноотстоящих узлов**. Это — единственный многочлен, решающий задачу интерполяции<sup>1)</sup>.



**2. Интерполяционный многочлен Лагранжа для равноотстоящих узлов.** Рассмотрим случай, когда значения  $x_i$  являются равноотстоящими, т. е.

$$x_1 - x_0 = x_2 - x_1 = \dots = x_n - x_{n-1} = h.$$

При этом, если ввести обозначение  $\frac{x - x_0}{h} = t$ , то получим:

$$\begin{aligned} \Phi_i(x) &= \frac{(x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} = \\ &= \frac{th(th - h) \dots [th - (i-1)h][th - (i+1)h] \dots (th - nh)}{ih(i-1)h \dots h(-h) \dots [-(n-i)h]} = \\ &= \frac{t(t-1) \dots (t-n)}{t-i} \frac{(-1)^{n-i}}{i!(n-i)!} = (-1)^{n-i} C_n^i \frac{1}{t-i} \frac{t(t-1) \dots (t-n)}{n!}. \end{aligned}$$

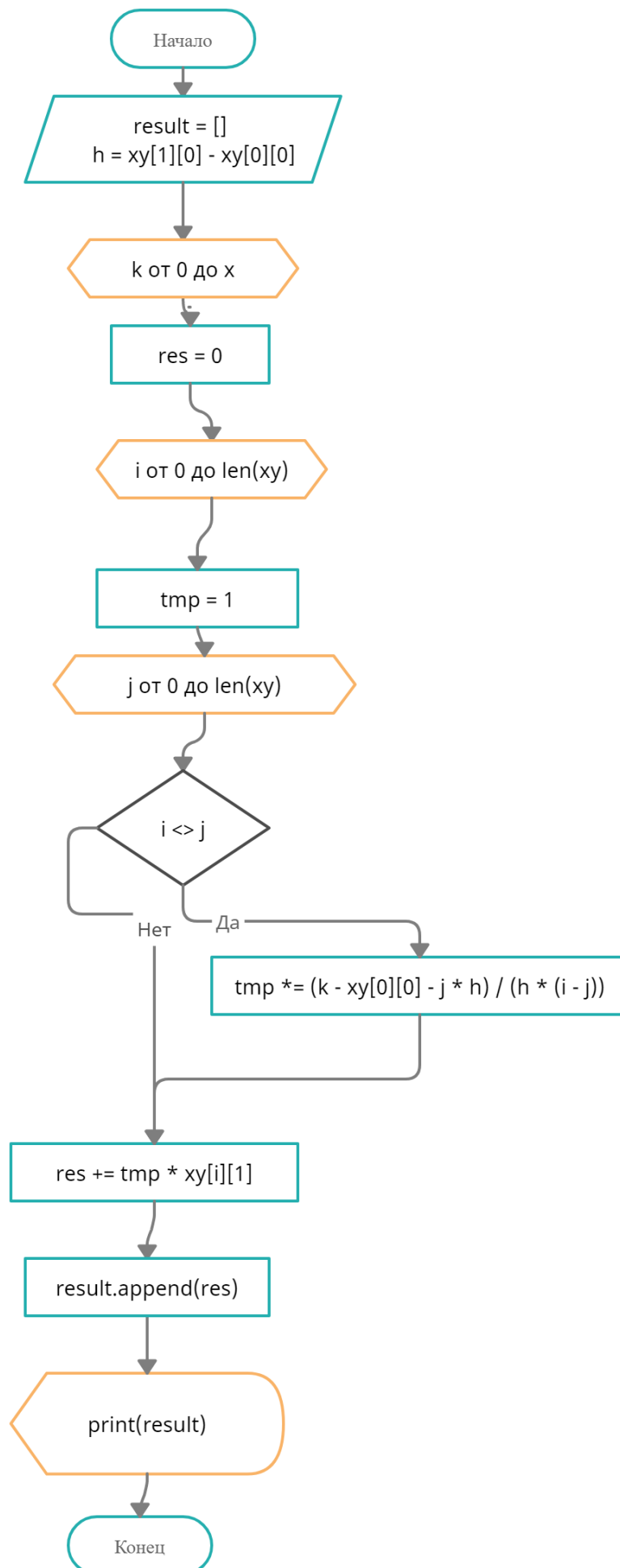
Итак,

$$\begin{aligned} L_n(x) &= L_n(x_0 + th) = \\ &= (-1)^n \frac{t(t-1) \dots (t-n)}{n!} \sum_{i=0}^n (-1)^i \frac{C_n^i y_i}{t-i}. \end{aligned} \quad (4)$$

Здесь и в дальнейшем для сокращения записей мы будем обозначать  $f(x_i)$  через  $y_i$ . В последнем выражении коэффициенты, стоящие перед  $y_i$ :

$$(-1)^{n-i} C_n^i \frac{t(t-1) \dots (t-n)}{(t-i)n!},$$

не зависят ни от функции  $f(x)$ , ни от  $h$  — шага таблицы. Их можно табулировать и использовать в самых различных случаях. Такие таблицы составлены и известны под названием таблиц *коэффициентов Лагранжа*.



## Расчетные данные

### Задание №1

x	y	№ варианта	Значение аргумента				
			$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
0,01	0,991824	4	0,11	0,13	0,62	0,005	0,44
0,06	0,951935						
0,11	0,91365						
0,16	0,876905						
0,21	0,841638						
0,26	0,807789						
0,31	0,775301						
0,36	0,74412						
0,41	0,714193						
0,46	0,68547						
0,51	0,657902						
0,56	0,631442						
0,61	0,611242						

Значения, полученные при помощи многочлена Ньютона для равноотстоящих узлов:

X	Y
0,11	0.914
0,13	0.898
0,62	0.608
0,005	0.996
0,44	0.674

Значения, полученные при помощи многочлена Лагранжа для равноотстоящих узлов:

X	Y
0,11	0.914
0,13	0.899
0,62	0.610

0,005	0.996
0,44	0.697

Задание №2

x	y	№ варианта	$x_1$	$x_2$
0.41	2.57418	4	0.616	0.444
0.46	2.32513			
0.52	2.09336			
0.60	1.86208			
0.65	1.74926			
0.72	1.62098			

Значения, полученные при помощи многочлена Лагранжа для  
неравноотстоящих узлов:

X	Y
0.616	1.824
0.444	2.398

## Листинг разработанной программы

### Main.py

```
import solutionMethods as sm

# x и y из 1 задания
xy1 = [[0.15, 4.4817], [0.16, 4.953], [0.17, 5.4739], [0.18, 6.0496], [0.19, 6.6859], [0.20, 7.3891], [0.21, 8.1662], [0.22, 9.025], [0.23, 9.9742], [0.24, 11.0232], [0.25, 12.1825], [0.26, 13.4637], [0.27, 13.5123]]
xi = [0.166, 0.266, 0.277, 0.144, 0.22] # Точки, в которых нужно найти значения функции в первом задании

xy2 = [[0.43, 1.63597], [0.48, 1.73234], [0.55, 1.87686], [0.62, 2.03345], [0.70, 2.22846], [0.75, 2.35973]] # x и y из 2 задания
xj = [0.512, 0.441] # Точки, в которых нужно найти значения функции во втором задании

sm.newtonPolynomial(xy1, xi)
sm.lagrangEquitable(xy1, xi)
sm.lagrangUnequitable(xy2, xj)
```

### solutionMethods.py

```
import math
import os
import keyboard

def newtonPolynomial(xy, x):
    os.system('cls')
    print("Многочлен Ньютона:")
    temp = [] # Временные значения
    dy = [] # Конечные разности
    result = [] # Список результатов
    for i in range(len(xy) - 1): # Конечные разности первого порядка заносим во временный список
        temp.append(xy[i + 1][1] - xy[i][1]) # Вычисляем сами разности
        dy.append(temp) # Временный список заносим в список конечных разностей

    for i in range(len(xy) - 2): # На каждом новом шаге вычисляем конечные разности следующих порядков
        temp = [] # Очищаем временные значения
        for j in range(len(dy[i]) - 1):
            temp.append(dy[i][j + 1] - dy[i][j]) # Вычисляем конечные разности
        dy.append(temp) # Временный список заносим в список конечных разностей

    h = xy[1][0] - xy[0][0] # Вычисляем шаг
    middle = (xy[0][0] + xy[len(xy) - 1][0]) / 2 # Считаем середину отрезка иксов

    for k in x: # Для каждой точки, в которой нужно найти значение, находим это значение
```

```

        if k < middle: # Если  $x_i$  лежит в промежутке от  $x_0$  до  $(x_0 + x_n) / 2$ 
            t = (k - xy[0][0]) / h # Вычисляем  $t$  по формуле  $(x - x_0)/h$ 
            res = 0 # Интерполяция вперед
            res += xy[0][1] # Прибавляем к результату  $y_0 + t * dy_0$ 
            res += t * dy[0][0]
            for i in range(1, len(dy)): # Считаем остальное:  $(t * (t-1) * \dots * (t - n + 1) * dny_0) / n!$ 
                tmp = 1
                for j in range(1, i): # Для удобства отдельно считаем  $(t - 1) \dots (t - n + 1)$ 
                    tmp *= (t - j)
                res += tmp * t * dy[i][0] / math.factorial(i + 1) # Добавляем  $(t * (t-1) * \dots * (t - n + 1) * dny_0) / n!$ 
            result.append(res)
        else:
            t = (k - xy[len(xy) - 1][0]) / h # Вычисляем  $t$  по формуле  $(x - x_0)/h$ 
            res = 0 # Интерполяция назад
            res += xy[len(xy) - 1][1] # Прибавляем к результату  $y_0 + t * dy_0$ 
            res += t * dy[0][len(dy[0])-1]
            for i in range(1, len(dy)): # Считаем остальное:  $(t * (t-1) * \dots * (t - n + 1) * dny_0) / n!$ 
                tmp = 1
                for j in range(1, i): # Для удобства отдельно считаем  $(t - 1) \dots (t - n + 1)$ 
                    tmp *= (t + j)
                res += t * tmp * dy[i][len(dy[i])-1] / math.factorial(i+1) # Добавляем  $(t * (t-1) * \dots * (t - n + 1) * dny_0) / n!$ 
            result.append(res)
    print(result)
    keyboard.wait('\n')
    os.system('cls')

def lagrangEquitable(xy, x):
    print("Лагранж равноотстоящий:")
    result = [] # Список результатов
    h = xy[1][0] - xy[0][0] # Считаем шаг
    for k in x: # Для каждой точки, в которой нужно найти значение, находим это значение
        res = 0
        for i in range(len(xy)):
            tmp = 1 # Временный буфер
            for j in range(len(xy)):
                if i != j:
                    tmp *= (k - xy[0][0] - j * h) / (h * (i - j)) # Считаем  $(X - X_i - j * h) / (h(i - j))$ 
            res += tmp * xy[i][1]
        result.append(res) # Заносим в список результатов

```



```

print(result)
keyboard.wait('\n')
os.system('cls')

def lagrangUnequitable(xy, x):
    print("Лагранж неравноотстоящий:")
    result = [] # Список результатов
    for k in x: # Для каждой точки, в которой нужно найти значение, находим это з
начение
        res = 0
        for i in range(len(xy)):
            tmp = 1 # Временный буфер
            for j in range(len(xy)):
                if i != j:
                    tmp *= (k - xy[j][0]) / (xy[i][0] - xy[j][0]) # Считаем (X -
Xj) / (Xi - Xj)
            res += tmp * xy[i][1]
        result.append(res) # Заносим в список результатов
    print(result)
    keyboard.wait('\n')
    os.system('cls')

```

## Результаты работы программы

Многочлен Ньютона:

[0.9136286666666666, 0.8979796316030974, 0.6084120276684258, 0.9957265824220307, 0.6740367318975928]

Лагранж равноотстоящий:

[0.91365, 0.898769351473783, 0.6104727149936899, 0.9959584504240641, 0.6968163503580965]

Лагранж неравноотстоящий:

## Вывод

В ходе данной работы были закреплены знания и умения по интерполированию функции с помощью многочленов Ньютона и Лагранжа.