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](#_Toc68555013)

[Постановка задачи 4](#_Toc68555014)

[Теоретические сведения 6](#_Toc68555015)

[Многочлен Ньютона 6](#_Toc68555016)

[Многочлен Лагранжа для неравноотстоящих узлов 9](#_Toc68555017)

[Многочлен Лагранжа для равноотстоящих узлов 11](#_Toc68555018)

[Расчетные данные 13](#_Toc68555019)

[Листинг разработанной программы 15](#_Toc68555020)

[Результаты работы программы 18](#_Toc68555021)

[Вывод 19](#_Toc68555022)

# 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 | | | | |
|  |  |  |  |  |
| 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 |  |  |
| 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 |  |  |  |

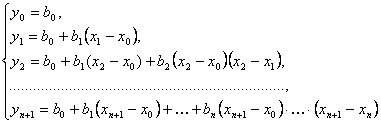
# Теоретические сведения

# Многочлен Ньютона

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

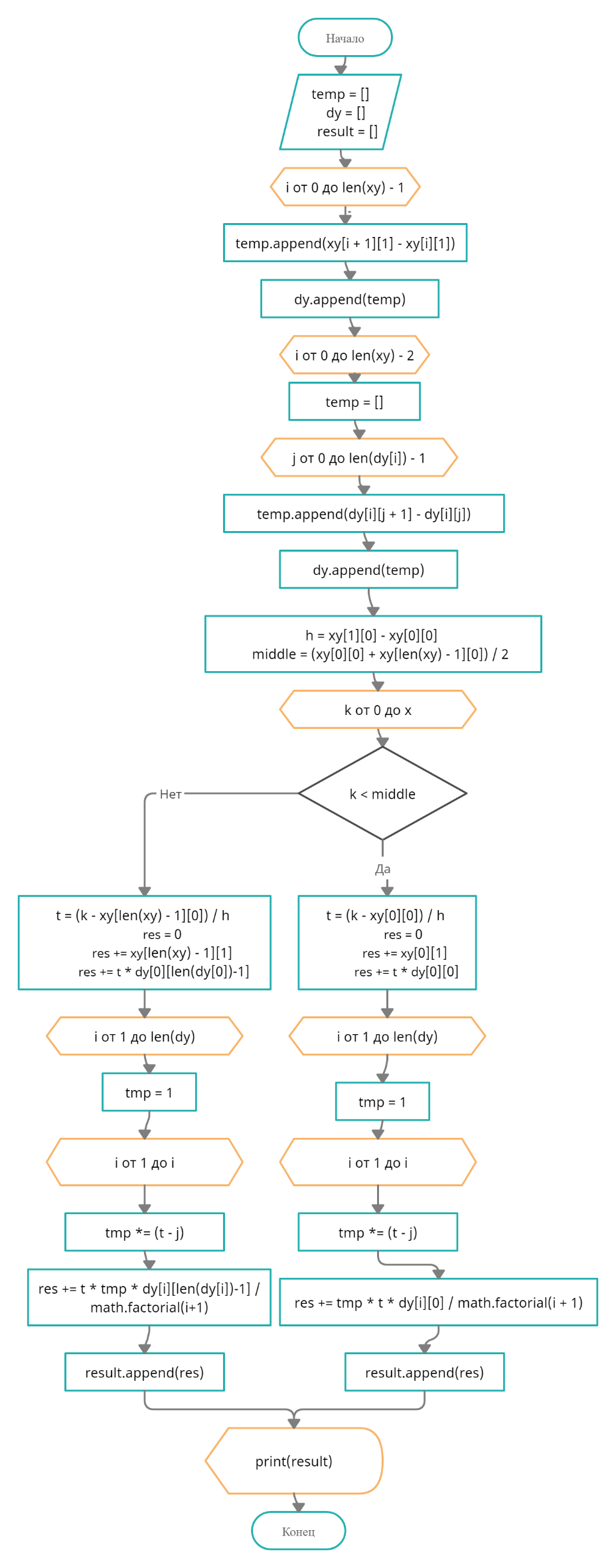
Пусть в точках  х0, х1,…, хn+1  значения  функции  у = f(x)  равны  соответственно у0 = f(x0), y1 = f(x1), …, yn+1 = f(xn+1).

Построим интерполяционный многочлен Ньютона с помощью метода неопределенных коэффициентов. Для этого запишем искомый многочлен в виде  
Pn(x) = b0 + b1(x – x0) + b2(x – x0)(x – x1) + b3(x – x0)(x – x1)(x – x2) + … + bn(x – x0)…(x – xn). (1)

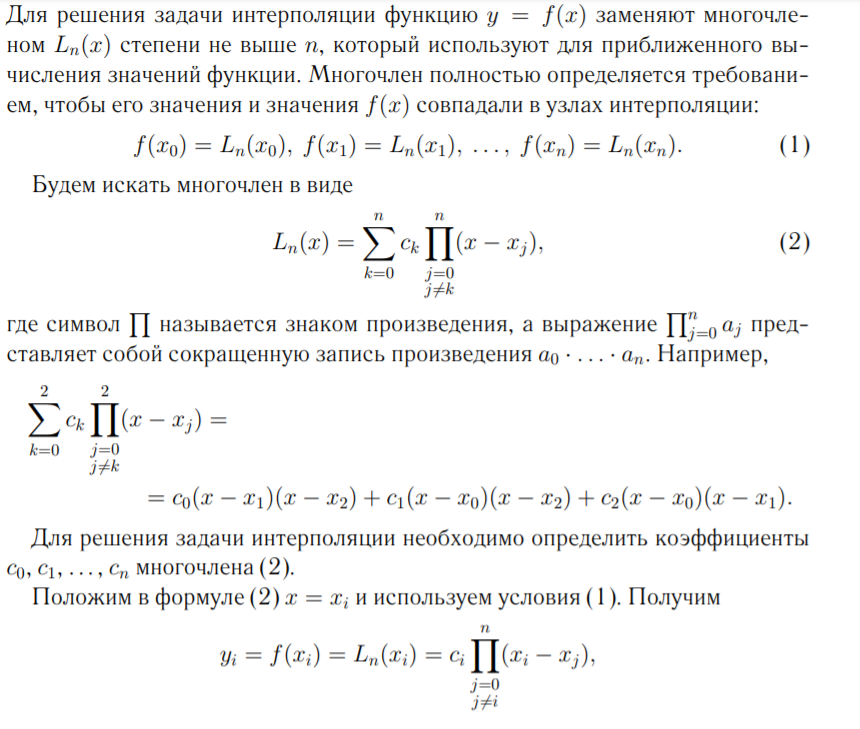
Последовательно подставляя в формулу (1) вместо х данные значения х0, х1, ..., хn+1, получим для нахождения неопределенных коэффициентов b0, b1, ..., bn «треугольную» систему уравнений  
  
(при подстановке в равенство (1) вместо х числа х0 в правой части равенства обратились в нуль все слагаемые, кроме первого: там везде был множитель (х – х0), обратившийся  в нуль; при подстановке х = х1 обратились в нуль все слагаемые, кроме первого и второго – они содержат множитель (х – х1) и т.д.).

Полученную систему удобно решать: из первого её уравнения находим свободный член искомого многочлена b0; подставив его во второе уравнение, находим коэффициент  b1 при первой степени х в искомом многочлене:  
http://school-collection.edu.ru/dlrstore-wrapper/f38c88e8-8328-443f-b1e8-d215a9dd734d/image004.gif  
и т.д.

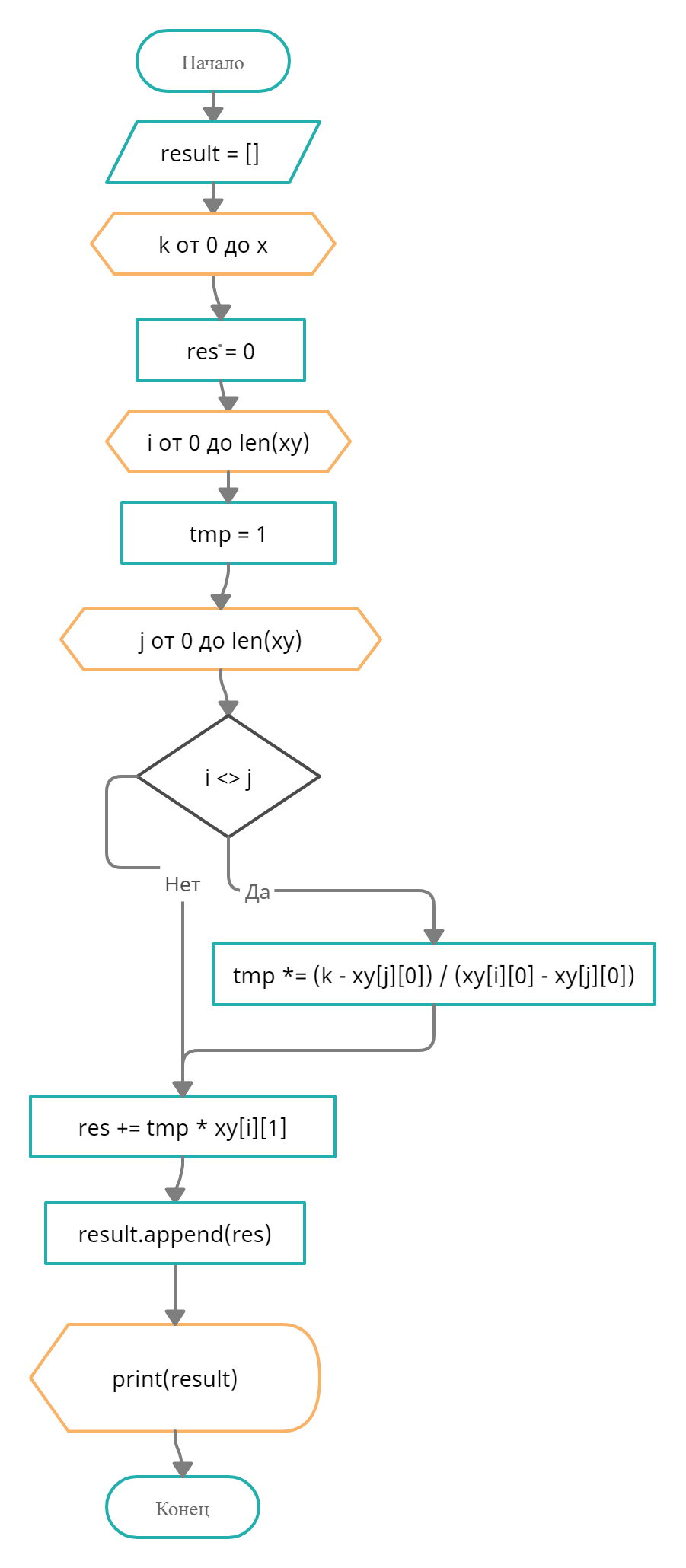
Для интерполяционного многочлена Ньютона можно выписать явные выражения коэффициентов через данные задачи, а также и оценки точности замены неизвестной функции f(x) этим многочленом.



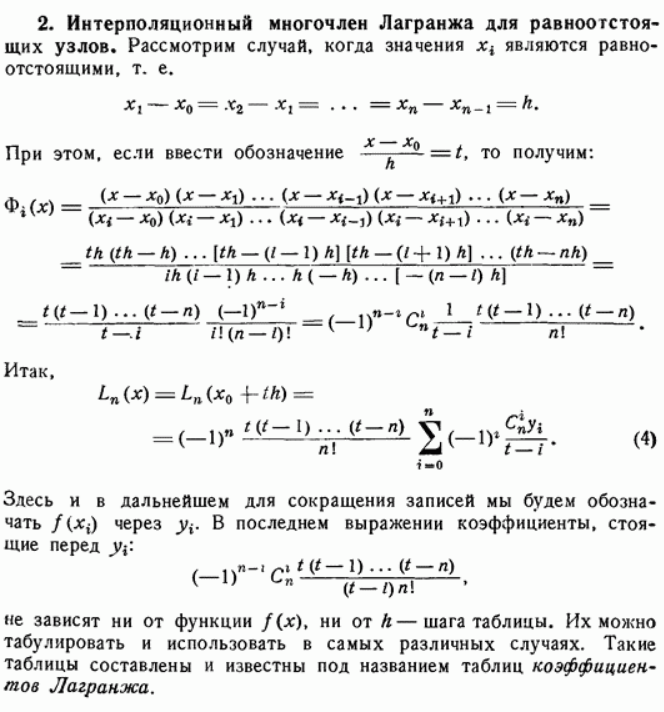
# Многочлен Лагранжа для неравноотстоящих узлов

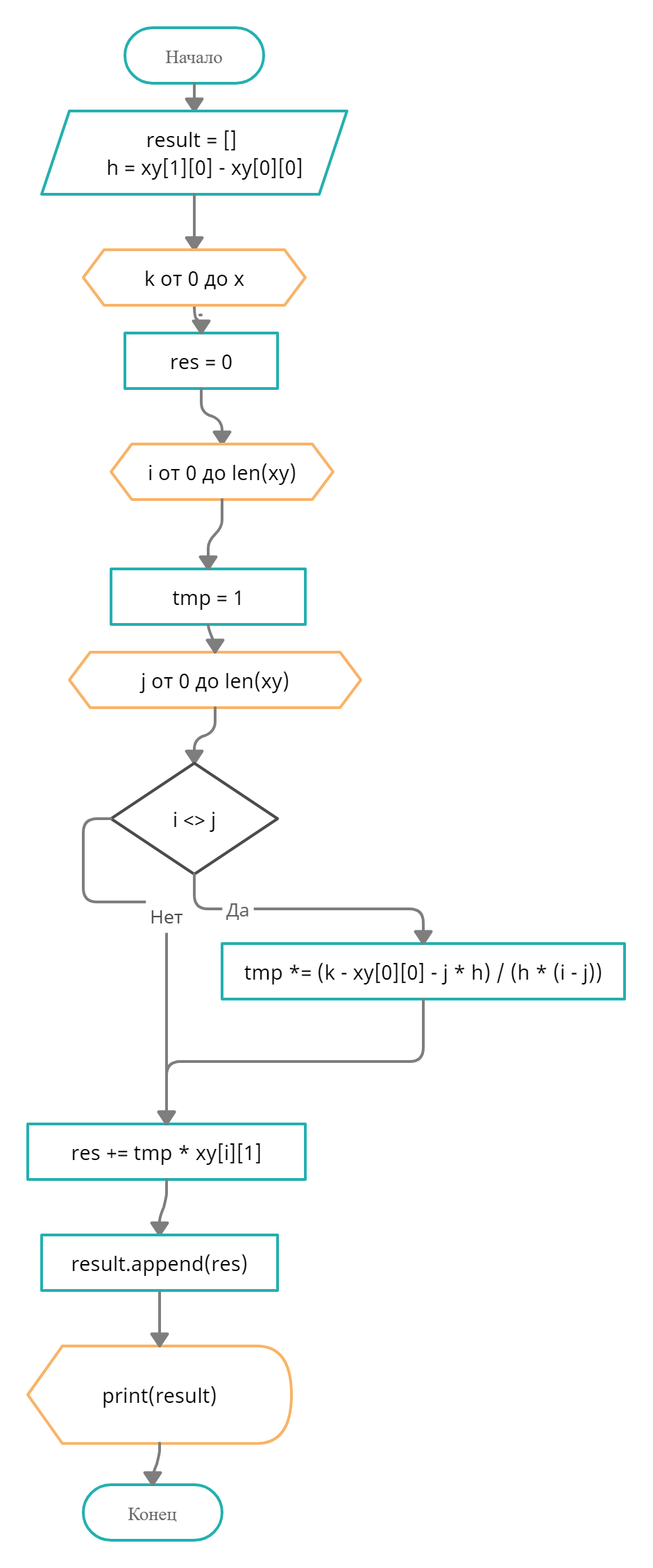






# Многочлен Лагранжа для равноотстоящих узлов





# Расчетные данные

Задание №1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | y | № варианта | Значение аргумента | | | | |
|  |  |  |  |  |
| 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:

|  |  |
| --- | --- |
| Х | У |
| 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:

|  |  |
| --- | --- |
| Х | У |
| 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 |  |  |
| 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:

|  |  |
| --- | --- |
| Х | У |
| 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 xi лежит в промежутке от x0 до (x0 + xn) / 2

            t = (k - xy[0][0]) / h # Вычисляем t по формуле (x - x0)/h

            res = 0  # Интерполяция вперед

            res += xy[0][1] # Прибавляем к результату  y0 + t \* dy0

            res += t \* dy[0][0]

            for i in range(1, len(dy)): # Считаем остальное:  (t \* (t-1) \* ...  \* (t - n + 1) \* dny0) / n!

                tmp = 1

                for j in range(1, i): # Для удобства отдельно считаем (t - 1)..(t - n+1)

                    tmp \*= (t - j)

                res += tmp \* t \* dy[i][0] / math.factorial(i + 1) # Добавляем  (t \* (t-1) \* ...  \* (t - n + 1) \* dny0) / n!

            result.append(res)

        else:

            t = (k - xy[len(xy) - 1][0]) / h # Вычисляем t по формуле (x - x0)/h

            res = 0 # Интерполяция назад

            res += xy[len(xy) - 1][1] # Прибавляем к результату  y0 + t \* dy0

            res += t \* dy[0][len(dy[0])-1]

            for i in range(1, len(dy)): # Считаем остальное:  (t \* (t-1) \* ...  \* (t - n + 1) \* dny0) / n!

                tmp = 1

                for j in range(1, i): # Для удобства отдельно считаем (t - 1)..(t - n+1)

                    tmp \*= (t + j)

                res += t \* tmp \* dy[i][len(dy[i])-1] / math.factorial(i+1) # Добавляем  (t \* (t-1) \* ...  \* (t - n + 1) \* dny0) / 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 (Х - Хi - 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 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[j][0]) / (xy[i][0] - xy[j][0]) # Считаем (X - Xj) / (Xi - Xj)

            res += tmp \* xy[i][1]

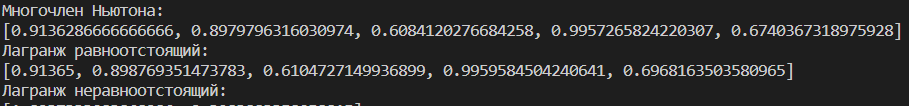
        result.append(res) # Заносим в список результатов

    print(result)

    keyboard.wait('\n')

    os.system('cls')

# Результаты работы программы



# Вывод

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