|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 1**

|  |  |
| --- | --- |
| **Тема «Алгоритм и программа построения интерполяционного полинома Ньютона»**  **Студент Кузин Антон**  **Группа ИУ7-42Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель Градов Владимир Михайлович** |  |

Москва.

2020 г.

**Цель работы:**

Целью данной лабораторной работы было реализовать алгоритм \*интерполирования\*.

Изначально выбирается конфигурация, на данном этапе проверяется, находится ли x в таблице, в противном случае выводится сообщение, о том, что экстраполяция запрещена.

После чего в отдельной функции рассчитываются разделенные разности, записываемые в массивы, которые в дальнейшем используются в качестве коэффициентов интерполяционного полинома Ньютона. Полином с полученными коэффициентами используется для вычисления значения функции от заданного X.

**Входные данные:** пункт меню и необходимые аргументы; текстовый файл, содержащий значения функции.

**Выходные данные:** сообщение о невозможности получения результата, либо результат интерполяции.

def read(filename):

file = open(filename, "r")

x = []

y = []

while 1:

line = file.readline()

if line == '':

break

nums = line.split()

x.append(float(nums[0]))

y.append(float(nums[1]))

file.close()

return x, y

def chose\_configuration(table, x, n):

xpos = -1

for i in range(1, len(table[0])): # поиск позиции x

if table[0][i] > table[0][i - 1]:

if table[0][i] >= x > table[0][i - 1]:

xpos = i

break

if table[0][i] < table[0][i - 1]:

if table[0][i] <= x < table[0][i - 1]:

xpos = i

break

if xpos == -1: # x вне таблицы

return None, None

left\_border = xpos - (n // 2) - 1

right\_border = xpos + (n // 2 + n % 2)

if left\_border < 0:

left\_border = 0

right\_border = n + 1

elif right\_border > len(table[1]):

left\_border = len(table[1]) - n

right\_border = len(table[1])

work\_array\_x = []

poly\_coefs = [[0 for j in range(n - i + 1)] for i in range(n + 1)]

for i in range(left\_border, right\_border):

work\_array\_x.append(table[0][i])

poly\_coefs[0][i - left\_border] = table[1][i]

return work\_array\_x, poly\_coefs

def calculate\_polynomial(work\_array\_x, poly\_coefs, x, n):

# готовим массивы для коеффициентов полинома, начиная с y из таблицы

for i in range(1, n + 1):

for j in range(n - i + 1):

poly\_coefs[i][j] = (poly\_coefs[i - 1][j + 1] - poly\_coefs[i - 1][j]) / (work\_array\_x[j + i] - work\_array\_x[j])

res = poly\_coefs[0][0]

mult = 1

for i in range(n):

mult \*= (x - work\_array\_x[i])

res += mult \* poly\_coefs[i + 1][0]

return res

def interpolate(table, x, n):

work\_array\_x, poly\_coefs = chose\_configuration(table, x, n)

if work\_array\_x is None:

return None

return calculate\_polynomial(work\_array\_x, poly\_coefs, x, n)

def find\_root\_interpolate(table, n):

t\_x = [table[1][i] for i in range(len(table[0]))]

t\_y = [table[0][i] for i in range(len(table[0]))]

work\_array\_x, poly\_coefs = chose\_configuration([t\_x, t\_y], 0, n)

if work\_array\_x is None:

for i in range(len(table[0])):

table[0][i] = table[1][i]

table[1][i] = t[i]

return None

res = calculate\_polynomial(work\_array\_x, poly\_coefs, 0, 4)

return res

def bisection(table):

r\_x = -1

r\_b\_n = -1

for i in range(len(table[0]) - 1):

for j in range(i + 1, len(table[0])):

if table[1][i] \* table[1][j] < 0:

r\_b\_n = j

break

if table[1][r\_b\_n] \* table[1][i] < 0:

r\_x = table[0][r\_b\_n]

break

if r\_b\_n == -1:

return None

l\_x = table[0][r\_b\_n - 1]

c = (r\_x + l\_x) / 2

n = 6

work\_array\_x, poly\_coefs = chose\_configuration(table, 0, n)

t = calculate\_polynomial(work\_array\_x, poly\_coefs, c, n)

t1 = calculate\_polynomial(work\_array\_x, poly\_coefs, r\_x, n)

t2 = calculate\_polynomial(work\_array\_x, poly\_coefs, l\_x, n)

if t1 == 0:

return r\_x

if t2 == 0:

return l\_x

prev\_c = c

rep = 0

eps = abs(t1 - t2) \* 0.000001

while abs(t) >= eps:

if abs(t1) < eps:

return r\_x

if abs(t2) < eps:

return l\_x

if t1 \* t < 0:

l\_x = c

elif t2 \* t < 0:

r\_x = c

c = (r\_x + l\_x) / 2

t = calculate\_polynomial(work\_array\_x, poly\_coefs, c, n)

t1 = calculate\_polynomial(work\_array\_x, poly\_coefs, r\_x, n)

t2 = calculate\_polynomial(work\_array\_x, poly\_coefs, l\_x, n)

if c == prev\_c:

rep += 1

else:

prev\_c = c

rep = 0

if rep >= 20:

return None

return c

def form\_new\_table():

name = input("Введите имя файла, в который будет сохранена таблица: ")

file = open(name, "w")

n = int(input("Введите количество значений: "))

print("Введите построчно таблицу в формате x y:")

for i in range(n):

file.write(input())

file.close()

def f(x):

return x \* x

def form\_new\_table\_f():

name = input("Введите имя файла, в который будет сохранена таблица: ")

file = open(name, "w")

i = float(input("Введите минимальный x: "))

x\_max = float(input("Введите максимальный x: "))

step = float(input("Введите шаг: "))

while i < x\_max:

file.write('%.3f' % (i) + ' ' + '%.3f'%(f(i)) + '\n')

i += step

file.close()

file = ''

table = []

while 1:

print("\n\nМеню: ")

print("1. Интерполяция")

print("2. Найти корень, используя метод половинного деления")

print("3. Найти корень, используя обратную интерполяцию")

print("4. Выбрать файл")

print("5. Сформировать новый файл вручную")

print("6. Сформировать файл по заданной функции")

print("7. Выход")

ch = int(input("\nВаш выбор: "))

if ch < 0 or ch > 7:

pass

elif ch == 1:

if len(file) == 0:

print("Файл не выбран")

continue

if len(table) == 0:

table = read(file)

x = float(input("Введите X для интерполяции: "))

n = int(input("Введите степень полинома: "))

res = interpolate(table, x, n)

if res is None:

print("Экстраполяция запрещена")

continue

print("Результат интерполяции: %.3f" % res)

elif ch == 2:

if file is '':

print("Файл не выбран")

continue

if len(table) == 0:

table = read(file)

res = bisection(table)

if res is None:

print("Корней нет, либо их не удалось найти")

continue

print("Корень: %.3f" % res)

elif ch == 3:

if file is '':

print("Файл не выбран")

continue

if len(table) == 0:

table = read(file)

n = int(input("Введите степень полинома: "))

res = find\_root\_interpolate(table, n)

if res is None:

print("Корней нет, либо их не удалось найти")

continue

print("Корень: %.3f" % res)

elif ch == 4:

file = input("Введите название файла: ")

table = read(file)

elif ch == 5:

form\_new\_table()

elif ch == 6:

form\_new\_table\_f()

elif ch == 7:

break