**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 № 3**

**Тема** Построение и программная реализация алгоритма сплайн-интерполяции табличных функций.

**Студент** Пронин А.С.

**Группа** ИУ7-42Б

**Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Преподаватель** Градов В.М.

Москва.

2021 г

**Цель работы:** Получение навыков владения методами интерполяции таблично заданных функций с помощью кубических сплайнов.

# ****1. Исходные данные****

1. Таблица функции с количеством узлов N. Задать с помощью формулы в диапазоне [0..10] с шагом 1.

2. Значение аргумента x в первом интервале, например, при х=0.5 и в середине таблицы, например, при x= 5.5. Сравнить с точным значением. (Вводятся пользователем)

# 2. Код программы

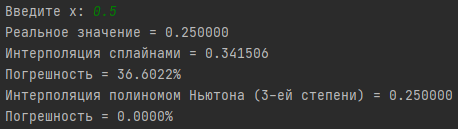
#Newton Interpolation  
from math import fabs, ceil  
  
def find\_beginning(x, table, table\_size, nearest\_value, degree):  
 degree += 1  
  
 if (nearest\_value == 0 and table[nearest\_value] > x):  
 print('Error!')  
 return 0  
  
 if (nearest\_value == table\_size - 1 and table[nearest\_value] < x):  
 print('Error!')  
 return table\_size - degree - 1  
  
 if (x <= table[nearest\_value]):  
 if (nearest\_value < (degree) / 2):  
 return 0  
 if ((table\_size - 1 - nearest\_value) < (ceil(degree / 2) - 1)):  
 return table\_size - degree - 1  
 return nearest\_value - degree / 2  
  
 if (x > table[nearest\_value]):  
 if (nearest\_value < (ceil(degree / 2) - 1)):  
 return 0  
 if (table\_size - 1 - degree < degree / 2):  
 return table\_size - 1 - degree  
 return nearest\_value - (ceil(degree / 2) - 1)  
 return 0  
  
def nearest\_value(x, table, size\_table):  
 if (x < table[0]):  
 return 0  
  
 if (x > table[size\_table - 1]):  
 return size\_table - 1  
  
 difference = fabs(x - table[0])  
 for\_first\_y = 0  
  
 for index in range(1, size\_table):  
 if (fabs(x - table[index]) < difference):  
 for\_first\_y = index  
 difference = fabs(x - table[index])  
  
 return for\_first\_y

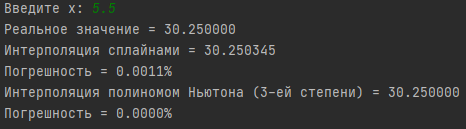
def newton\_polinom(x, degree, beginnig, table\_x, table\_y):  
 result = table\_y[beginnig]  
 for i in range((beginnig + 1), beginnig + degree):  
 divided = 0  
 for j in range(beginnig, i + 1):  
 difference = 1  
 for k in range(beginnig, i + 1):  
 if (k != j):  
 difference \*= (table\_x[j] - table\_x[k])  
 divided += (table\_y[j] / difference)  
 for k in range(beginnig, i):  
 divided \*= (x - table\_x[k])  
 result += divided  
 return result  
  
def newton\_interpolation(x, x\_table, y\_table, degree):  
 for\_first\_y = nearest\_value(x, x\_table, len(x\_table))  
 beginning = find\_beginning(x, x\_table, len(x\_table), for\_first\_y, degree)  
 return newton\_polinom(x, degree, int(beginning), x\_table, y\_table)  
#Spline Interpolation  
def build\_spline(x, y, n):  
 # Инициализация массива сплайнов  
 # [a, b, c, d, x]  
 splines = [[0, 0, 0, 0, 0] for \_ in range(0, n)]  
 for i in range(0, n):  
 splines[i][4] = x[i]  
 splines[i][0] = y[i]  
  
 splines[0][2] = splines[n - 1][2] = 0.0  
  
 # Решение СЛАУ относительно коэффициентов сплайнов c[i]   
 # Вычисление прогоночных коэффициентов - прямой ход метода прогонки  
 alpha = [0.0 for \_ in range(0, n - 1)]  
 beta = [0.0 for \_ in range(0, n - 1)]  
  
 for i in range(1, n - 1):  
 hi = x[i] - x[i - 1]  
 hi1 = x[i + 1] - x[i]  
 A = hi  
 C = 2.0 \* (hi + hi1)  
 B = hi1  
 F = 6.0 \* ((y[i + 1] - y[i]) / hi1 - (y[i] - y[i - 1]) / hi)  
 z = (A \* alpha[i - 1] + C)  
 alpha[i] = -B / z  
 beta[i] = (F - A \* beta[i - 1]) / z  
  
 # Нахождение решения - обратный ход метода прогонки  
 for i in range(n - 2, 0, -1):  
 splines[i][2] = alpha[i] \* splines[i + 1][2] + beta[i]  
  
 # По известным коэффициентам c[i] находим значения b[i] и d[i]  
 for i in range(n - 1, 0, -1):  
 hi = x[i] - x[i - 1]  
 splines[i][3] = (splines[i][2] - splines[i - 1][2]) / hi  
 splines[i][1] = hi \* (2.0 \* splines[i][2] + splines[i - 1][2]) / 6.0 + (y[i] - y[i - 1]) / hi  
 #print(splines)  
 return splines  
  
# Вычисление значения интерполированной функции в произвольной точке  
def spline\_interpolation(splines, x):  
 n = len(splines)  
 # [a, b, c, d, x]  
 s = [0, 0, 0, 0, 0]  
  
 if x <= splines[0][4]:  
 s = splines[0]  
 elif x >= splines[n - 1][4]:  
 s = splines[n - 1]  
 else:  
 i = 0  
 j = n - 1  
 while i + 1 < j:  
 k = i + (j - i) // 2  
 if x <= splines[k][4]:  
 j = k  
 else:  
 i = k  
 s = splines[j]  
  
 dx = x - s[4]  
 #print(s)  
 # Вычисляем значение сплайна в заданной точке по схеме Горнера  
 return s[0] + (s[1] + (s[2] / 2.0 + s[3] \* dx / 6.0) \* dx) \* dx;  
  
  
#Main Program  
def function(x):  
 y = x\*\*2  
 return y  
  
def main():  
 x\_table = []  
 y\_table = []  
 for i in range(11):  
 x\_table.append(i)  
 y\_table.append(function(i))  
  
 x = float(input("Введите x: "))  
 spline = build\_spline(x\_table, y\_table, len(y\_table))  
 real\_res = function(x)  
 spline\_res = spline\_interpolation(spline, x)  
 newton\_res = newton\_interpolation(x, x\_table, y\_table, 3)  
 print("Реальное значение = {:^.6f}".format(real\_res))  
 print("Интерполяция сплайнами = {:^.6f}".format(spline\_res))  
 print("Погрешность = {:.4f}%".format((fabs(spline\_res - real\_res) / fabs(real\_res)) \* 100))  
 print("Интерполяция полиномом Ньютона (3-ей степени) = {:^.6f}".format(newton\_res))  
 print("Погрешность = {:.4f}%".format((fabs(newton\_res - real\_res) / fabs(real\_res)) \* 100))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# 3. Результаты работы

1. Значения y(x).

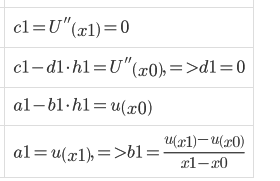
2. Сравнить результаты интерполяции кубическим сплайном и полиномом Ньютона 3-ей степени.





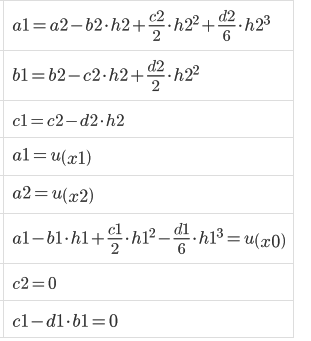
# 4. Вопросы при защите лабораторной работы

***1. Получить выражения для коэффициентов кубического сплайна, построенного на двух точках.***



Полином выродился в прямую.

***2. Выписать все условия для определения коэффициентов сплайна, построенного на 3-х точках.***

******

***3. Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо C1=C2.***



При => => =>

***4. Написать формулу для определения последнего коэффициента сплайна , чтобы можно было выполнить обратный ход метода прогонки, если в качестве граничного условия задано , где k,m и p - заданные числа.***



=>