Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
Санкт-Петербургский университет аэрокосмического приборостроения

КАФЕДРА № 2

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

доц., канд. техн. наук		С.Л. Козенко
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 3

Интерполяция

Вариант 5

по дисциплине: ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4128	05.04.2023	В.А. Воробьёв
		подпись, дата	инициалы, фамилия

Санкт – Петербург, 2023

СОДЕРЖАНИЕ

1 Цели и постановка задачи

1.1 Цели работы

а) освоение методов интерполяции функций;

б) совершенствование навыков по алгоритмизации и программированию вычислительных задач

1.2 Задание

Составить схему алгоритма и программу на языке C/C++ решения задачи по теме «Интерполяция» в соответствии с индивидуальным заданием.

№ варианта	Значения x_i					Значения y_i					Метод интерполяции
i	0	1	2	3	4	0	1	2	3	4	
5	0.3	0.5	0.7	0.9	1,1	1,2	0.7	0.3	-0.3	-1.4	Ньютона

Рисунок 1 – Вариант задания

2 Описание метода решения

Наиболее часто для определения функции φ(X) используется постановка, называемая постановкой задачи интерполяции.

В этой классической постановке задачи интерполяции требуется определить приближенную аналитическую функцию φ(X), значения которой в узловых точках Xi совпадают со значениями Y(Xi) исходной таблицы, т.е. условий

$$\boldsymbol{\varphi}(\boldsymbol{X}_i) = \boldsymbol{Y}_i \quad (\boldsymbol{i} = 0,1,2,\ldots,\boldsymbol{n})$$

Построенная таким образом аппроксимирующая функция φ(X) позволяет получить достаточно близкое приближение к интерполируемой функции Y(X) в пределах интервала значений аргумента [X0; Xn], определяемого таблицей. При задании значений аргумента X, не принадлежащих этому интервалу, задача интерполяции преобразуется в задачу экстраполяции. В этих случаях точность значений, получаемых при вычислении значений функции φ(X), зависит от расстояния значения аргумента X от X0 , если X < X0 , или от Xn , если X > Xn .

При математическом моделировании интерполирующая функция может быть использована для вычисления приближенных значений исследуемой функции в промежуточных точках подинтервалов [Xi; Xi+1]. Такая процедура называется <u>уплотнением таблицы</u>.

Алгоритм интерполяции определяется способом вычисления значений функции φ(X). Наиболее простым и очевидным вариантом реализации интерполирующей функции является замена исследуемой функции Y(X) на интервале [Xi; Xi+1] отрезком прямой, соединяющим точки Yi , Yi+1. Этот метод называется методом линейной интерполяции.

Интерполяционные многочлены Ньютона

Для интерполяции функций, заданных таблицами с равноотстоящими значениями аргумента

$$X_{i+1} - X_i = h \qquad (i = 0, 1, 2, \ldots, n-1)$$

построение интерполяционных формул и вычисление по этим формулам заметно упрощается. В записях этих интерполяционных алгоритмов используются разности между значениями функции в соседних узлах интерполяции.

Конечные разности

Конечной разностью первого порядка называется

$$\Delta y_i = (Y_{i+1} - Y_i) \quad (i = 0,1,2, \dots ,n-1).$$

Из конечных разностей первого порядка образуются конечные разности второго порядка:

$$\begin{aligned} \Delta^2 y_i &= \Delta y_{i+1} - \Delta y_i = (y_{i+2} - y_{i+1}) - (y_{i+1} - y_i) = \\ &= (y_{i+2} - 2y_{i+1} + y_i) \qquad\qquad (i = 0,1,2, \dots, n-2) \end{aligned}$$

Аналогично определяются конечные разности третьего, четвёртого и более высоких порядков.

Для вычисления конечных разностей обычно создаются таблицы, вид которых приводится ниже.

Таблица 1. Алгоритм построения конечных разностей

i	X	Y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
0	X_0	Y_0	$\Delta y_0 = Y_1 - Y_0$	$\Delta^2 y_0 = \Delta y_1 - \Delta y_0$	$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0$	$\Delta^4 y_0 = \Delta^3 y_1 - \Delta^3 y_0$
1	X_1	Y_1	$\Delta y_1 = Y_1 - Y_2$	$\Delta^2 y_1 = \Delta y_2 - \Delta y_1$	$\Delta^3 y_1 = \Delta^2 y_2 - \Delta^2 y_1$	
2	X_2	Y_2	$\Delta y_2 = Y_3 - Y_2$	$\Delta^2 y_2 = \Delta y_3 - \Delta y_2$		
3	X_3	Y_3	$\Delta y_3 = Y_4 - Y_3$			
4	X_4	Y_4				

Интерполяционные формулы Ньютона

Для функций, заданных таблицами с постоянным шагом изменения аргумента, наиболее часто используются первая или вторая формулы Ньютона, в которых интерполяционная функция определяется как многочлен вида:

$$P_n^{(I)}(x) = a_0 + a_1(x - X_0) + a_2(x - X_0)(x - X_1) + \dots + a_n(x - X_0)(x - X_1)\dots(x - X_{n-1})$$

при интерполяции от нулевого узла X0 или

$$P_n^{(II)}(x)=b_0+b_1(x-X_n)+b_2(x-X_n)(x-X_{n-1})+\ldots+b_n(x-X_n)(x-X_{n-1})\ldots(x-X_0)$$

при интерполяции от узла Xn.

Значения коэффициентов ai и bi в формулах (21) или (22) находятся из условий Лагранжа, определяющих в узлах интерполяции совпадение значений интерполирующей функции со значением таблично заданной функции:

$$P_n(x_i)=Y_i$$

Полагая X=X0 , в формуле (21) получим

$$Pn(X0)=a0=Y0\ .$$

Аналогично для X=X1

$$Pn(X1)=a0+a1(X1-X0)=Y1\ ,$$

и далее

$$a1=(Y1-Y0)/(X1-X0)$$

или, используя введённые обозначения,

$$a1=\Delta y0/(1!h).$$

Продолжая подстановки значений Xi , получим

$$Pn(X2)=a0+a1(X2-X0)+a2\,(X2-X0)(X2-X1)=Y2\ ,$$

и далее

$$a2*2h2=Y2-a0-a1*2h=Y2-Y0-\Delta y0/h*2h=Y2-2\,Y1+Y0=\Delta 2y0$$

откуда

$$a2*2h2=Y2-a0-a1*2h=Y2-Y0-\Delta y0/h*2h=Y2-2\,Y1+Y0=\Delta 2y0$$

Проведя аналогичные преобразования для X=X3 и X=X4, получим

$$a_2=\frac{\Delta^2 y_0}{2!\,h^2}\ .$$

3 Аналитические расчеты

В табличном процессоре Microsoft Excel реализовал алгоритм интерполяции по первой формуле Ньютона, заполнив первоначально ячейки исходными данными, соответствующими варианту. Далее основываясь на Таблице 1, заполнил таблицу промежуточных разностей. На основе этой таблицы произвел вычисления значений Y в интерполированных точках.

i	X	Y	Δy	Δ^2y	Δ^3y	Δ^4y
0	0,30	1,2	-0,5	0,1	-0,3	0,0
1	0,5	0,7	-0,4	-0,2	-0,3	
2	0,7	0,3	-0,6	-0,5		
3	0,9	-0,3	-1,1			
4	1,1	-1,4			0,20	
i	0	1	2	3	4	
h^i	1	0,20	0,04	0,01	0,00	
i!	1,0	1	2	6	24	
	a_0	a_1	a_2	a_3	a_4	
	1,2	-2,5	1,25	-6,25	0	
	0,7	-2	-2,5	-6,25		
	0,3	-3	-6,25			
	-0,3	-5,5				
	-1,4					

Рисунок 2 – Таблица Excel конечных разностей

Также по условию задания были выбраны 2 экстраполированные точки, которые будут включены в промежуток интерполирования. Экстраполированные точки были вычислены с помощью функции ПРЕДСКАЗ, а затем выделены серым цветом.

Таблица 2 - Интерполированные точки

i	X	Y_{interp}
0	**0,20**	**1,63**
1	**0,30**	**1,20**
2	**0,40**	**0,92**
3	**0,50**	**0,70**
4	**0,60**	**0,51**
5	**0,70**	**0,30**
6	**0,80**	**0,04**
7	**0,90**	**-0,30**
8	**1,00**	**-0,77**
9	**1,10**	**-1,40**
10	**1,20**	**-1,36**

4 Схема алгоритма решения задачи

Две важнейшие части нашего кода:

1) Функция divided_diff – отвечает за вычисление элемента в таблице конечных разностей для последующего использования в интерполировании методом Ньютона.

2) Функция newton_interpolation – высчитывает точки интерполяции методом Ньютона.

Опустим специфику ввода/вывода и изобразим блок-схемы для программного кода.

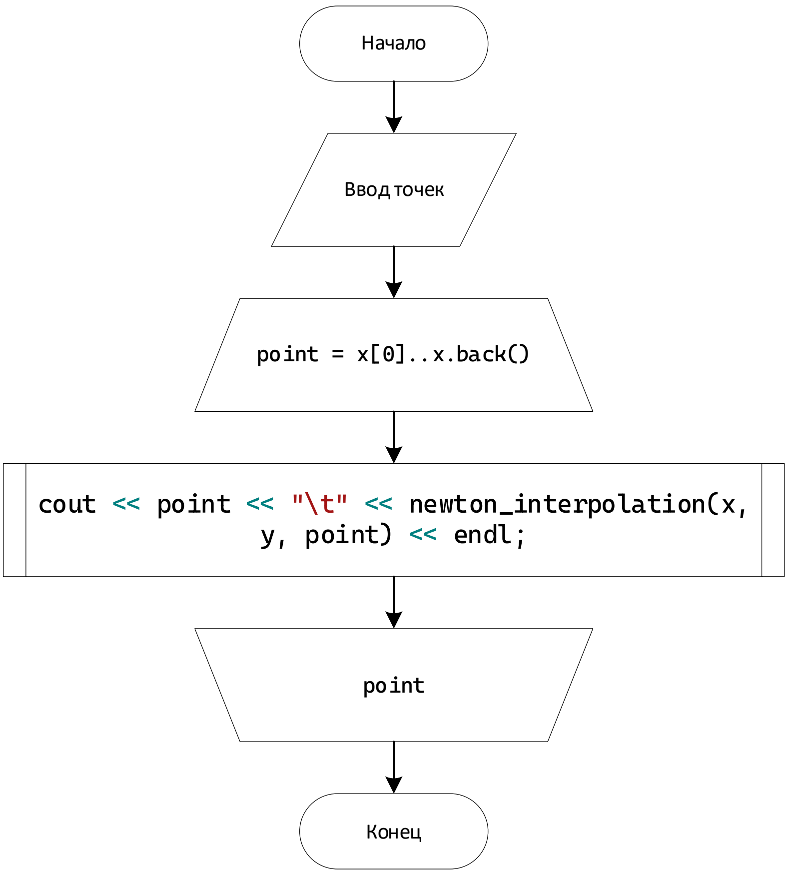


Рисунок 3 – Блок-схема main

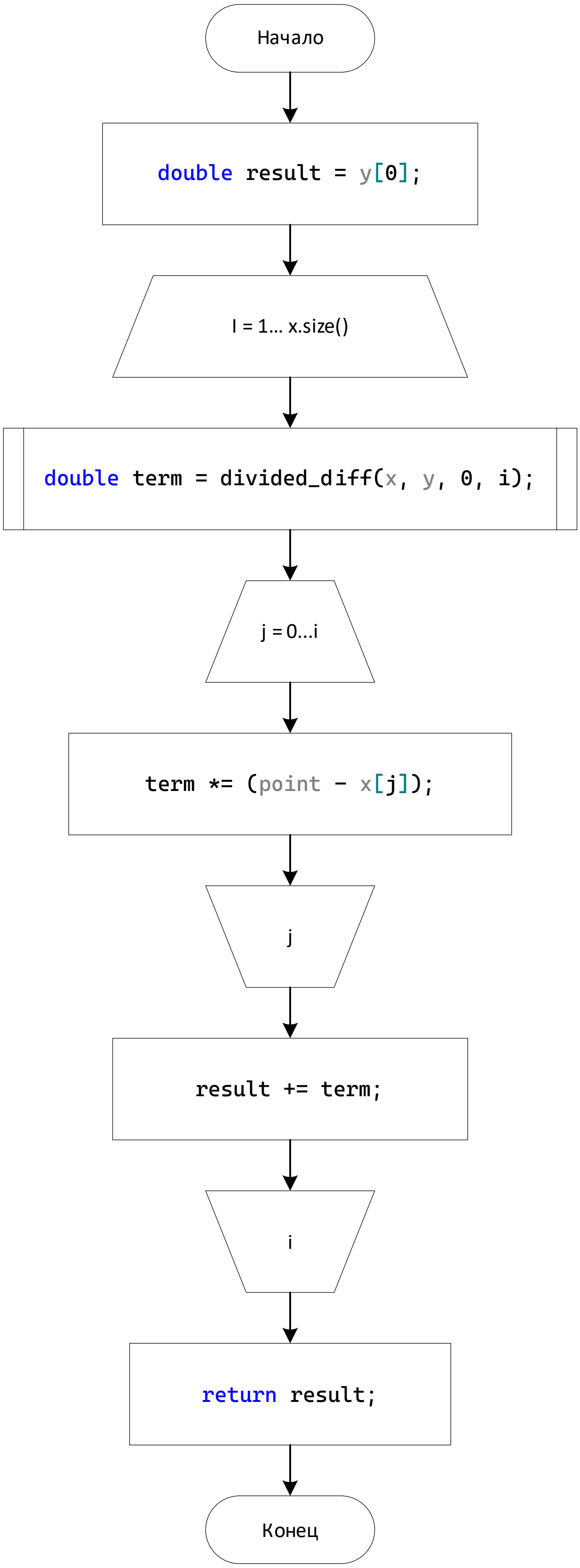


Рисунок 4 – Блок-схема newton_interpolation

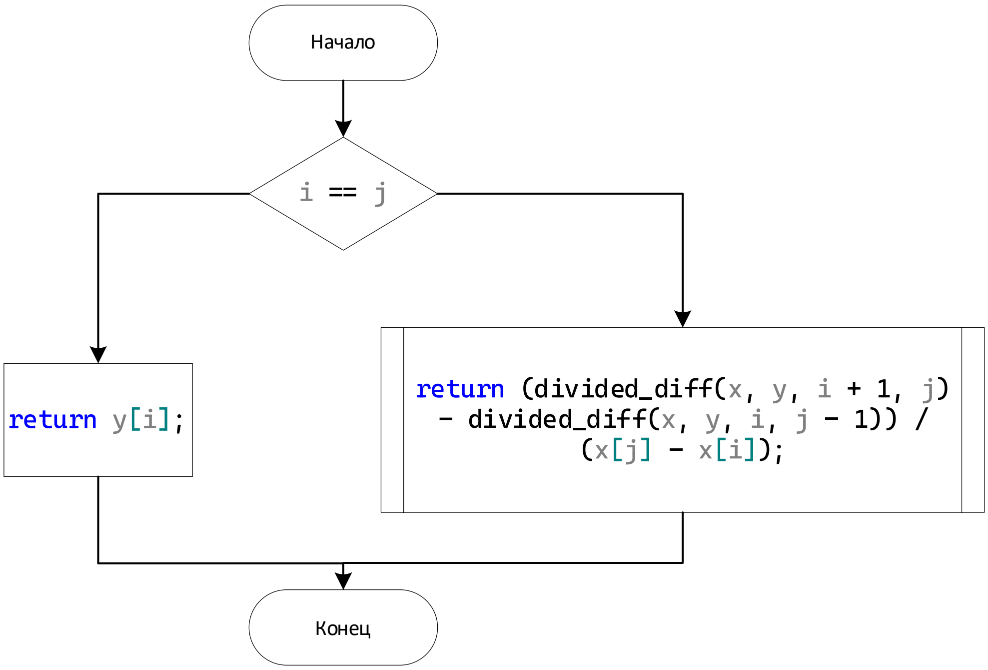


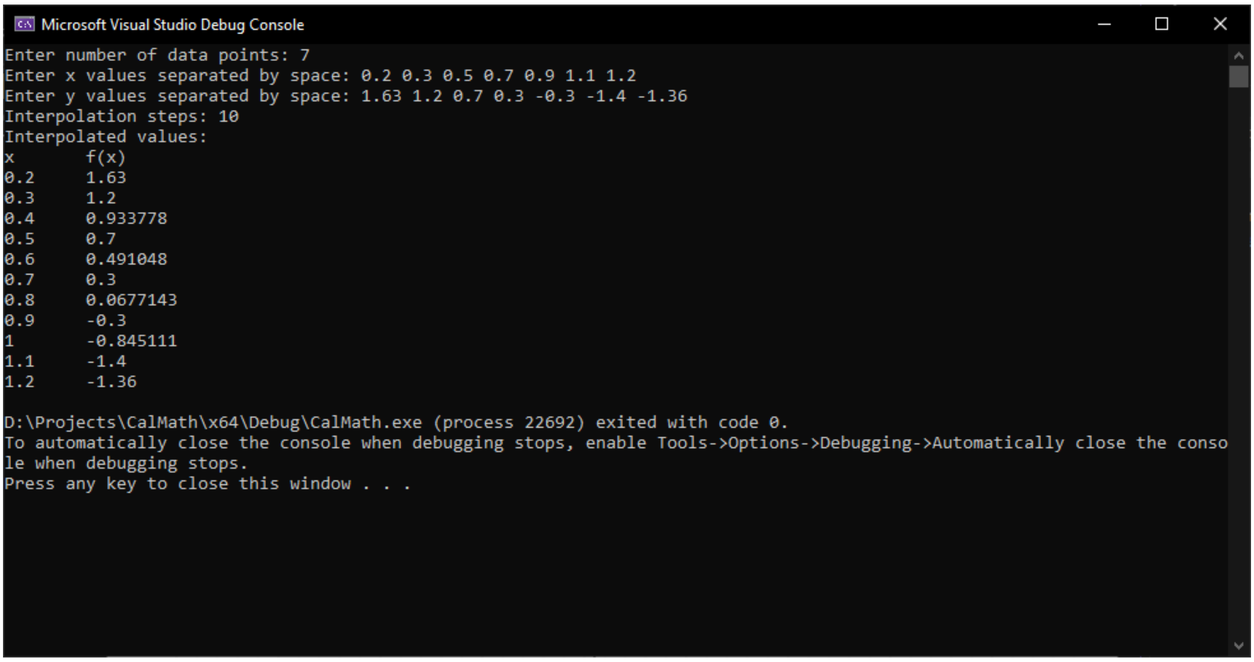
Рисунок 5 – Блок-схема divided_diff

5 Текст программы на C++

Исходный код доступен на GitHub, URL: https://github.com/vladcto/SUAI_homework/blob/df4c006204bfc3249fcfa463a18be9c686cd8b25/4_semester/CM/%D0%BB%D1%803/source.cpp

Также исходный код доступен в Приложении А.

6 Результаты программных расчетов



```
Microsoft Visual Studio Debug Console
Enter number of data points: 7
Enter x values separated by space: 0.2 0.3 0.5 0.7 0.9 1.1 1.2
Enter y values separated by space: 1.63 1.2 0.7 0.3 -0.3 -1.4 -1.36
Interpolation steps: 10
Interpolated values:
x       f(x)
0.2     1.63
0.3     1.2
0.4     0.933778
0.5     0.7
0.6     0.491048
0.7     0.3
0.8     0.0677143
0.9     -0.3
1       -0.845111
1.1     -1.4
1.2     -1.36

D:\Projects\CalMath\x64\Debug\CalMath.exe (process 22692) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Рисунок 6 – Результат выполнения программы

7 Сравнение программных и аналитических расчётов

На Таблице 3 представлены аналитические (Yinterp) и программные (Yprog) расчеты.

Таблица 3- Аналитические и программные расчеты

i	X	Y_{interp}	X	Yprog
0	**0,20**	1,63	**0,20**	1,63
1	**0,30**	1,20	**0,30**	1,20
2	**0,40**	0,92	**0,40**	0,93
3	**0,50**	0,70	**0,50**	0,70
4	**0,60**	0,51	**0,60**	0,49
5	**0,70**	0,30	**0,70**	0,30
6	**0,80**	0,04	**0,80**	0,01
7	**0,90**	-0,30	**0,90**	-0,30
8	**1,00**	-0,77	**1,00**	-0,85
9	**1,10**	-1,40	**1,10**	-1,40
10	**1,20**	-1,36	**1,20**	-1,36

Построим график для полученных значений и сделаем вывод.

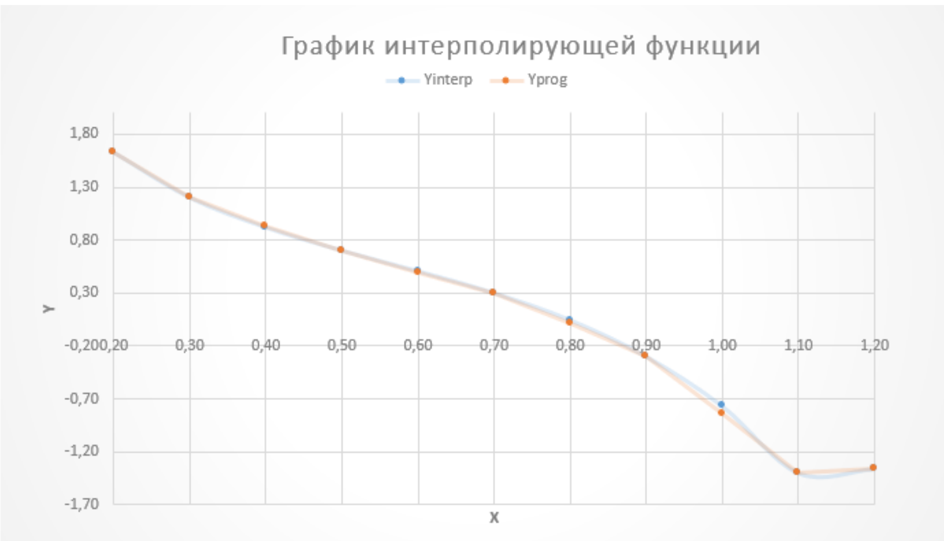


Рисунок 7 – График функций

Как мы видим, графики почти совпадают. Погрешности есть, но они вызваны накоплением математической ошибки при округлении чисел в Excel и C++. Также отметим, что написанная нами программа проходит точки для интерполяции без каких-либо погрешностей. Можем сделать вывод, что написанная нами программа является корректной реализацией алгоритма интерполяции Ньютона.

8 Выводы

В ходе выполнения лабораторной работы была составлена схема алгоритма и на её основе написана программа на языке C++. Написанная программа была сравнена с аналитическим расчетами, выполненными в Excel, на основе чего был сделан вывод о её корректности.

В результате выполнения лабораторной работы был освоен метод Ньютона для интерполяции функций, а также усовершенствованы навыки по алгоритмизации и программированию вычислительных задач.

Приложение А

Листинг программы

```
// 0.2 0.3 0.5 0.7 0.9 1.1 1.2
// 1.63 1.2 0.7 0.3 -0.3 -1.4 -1.36

#include <iostream>
#include <vector>

using namespace std;

double divided_diff(vector<double> x, vector<double> y, int i, int j) {
	if (i == j)
		return y[i];
	else
		return (divided_diff(x, y, i + 1, j) - divided_diff(x, y, i, j - 1)) /
(x[j] - x[i]);
}

double newton_interpolation(vector<double> x, vector<double> y, double point) {
	double result = y[0];
	for (int i = 1; i < x.size(); i++) {
		double term = divided_diff(x, y, 0, i);
		for (int j = 0; j < i; j++)
			term *= (point - x[j]);
		result += term;
	}
	return result;
}

int main() {
	int n;
	vector<double> x, y;
	double temp;

	cout << "Enter number of data points: ";
	cin >> n;

	cout << "Enter x values separated by space: ";
	for (int i = 0; i < n; i++) {
		cin >> temp;
		x.push_back(temp);
	}

	cout << "Enter y values separated by space: ";
	for (int i = 0; i < n; i++) {
		cin >> temp;
		y.push_back(temp);
	}

	double steps;
	cout << "Interpolation steps: ";
	cin >> steps;
	cout << "Interpolated values:\n";
	cout << "x\tf(x)\n";
	for (double point = x[0]; point <= x.back(); point += (x.back() - x[0]) /
steps) {
		cout << point << "\t" << newton_interpolation(x, y, point) << endl;
	}

	return 0;
}
```