Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

**Звіт**
Про виконання лабораторної роботи №8

**На тему:**

«наближення дискретних(таблично заданих функцій)»
з дисципліни «Чисельні методи»

**Лектор:**
доцент каф. ПЗ
Мельник Н. Б.
**Виконав:**
ст. гр. ПЗ-11
Морозов О. Р.
**Прийняла:**
доцент каф. ПЗ
Мельник Н. Б.

« __ » ________ 2022 р.

∑ = _____ _________ .

Львів – 2022

**Тема:** наближення дискретних(таблично заданих функцій)
**Мета:** ознайомитися з методом інтерполяції таблично заданих функцій.

## Теоретичні відомості

**Інтерполяційний поліном Лагранжа** – основна ідея цього методу полягає в пошуку полінома, який в одному довільному вузлі інтерполяції приймає значення одиниця, а в усіх інших вузлах - нуль. Наближену функцію y = F(x) розглянемо у вигляді:

$$F(x)=L_n(x)=\sum_{i=0}^{n}P_i(x)f(x_i)$$

Де $P_i(x)$ – такий многочлен, що $P_i(x_j)=\begin{cases}0, & i\neq j,\\ 1, & i=j,\end{cases}$ $\quad i,j=\overline{0,n}$

Оскільки точки x0, x1.. є коренями полінома то їх можна записати у такому вигляді:

$$P_i(x)=\frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}$$

А наближена функція F(x) матиме вигляд:

$$F(x)=L_n(x)=\sum_{i=0}^{n}\frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}f(x_i)$$

**Інтерполяційний поліном Ньютона** - цей спосіб полягає в тому, що поліном Pn(x) для загального випадку нерівновіддалених вузлів записують у вигляді:

$$P_n(x)=f(x_0)+f(x_0,x_1)(x-x_0)+f(x_0,x_1,x_2)(x-x_0)(x-x_1)+$$
$$+\dots+f(x_0,x_1,\dots,x_n)(x-x_0)(x-x_1)\dots(x-x_{n-1}),$$

Де $f(x_0,x_1,\dots,x_n)=\dfrac{f(x_1,\dots,x_n)-f(x_0,\dots,x_{n-1})}{x_n-x_0}$ - розділена різниця n-ого порядку.

Припустимо, що вузли інтерполяції є рівновіддаленими, тобто:

$$x_i=x_0+ih,\quad i=\overline{0,n}$$

Тоді скінченну різницю n-ого порядку запишемо у такому вигляді:

$$\Delta^n f(x_i)=\Delta(\Delta^{n-1}f(x_i))=\Delta^{n-1}f(x_{i+1})-\Delta^{n-1}f(x_i)$$

Підставивши скінченні різниці замість розділених різниць у інтерполяційну формулу Ньютона для нерівновіддалених вузлів отримаємо:

$$P_n(x) = f(x_0) + \frac{\Delta f(x_0)}{1!h}(x - x_0) + \frac{\Delta^2 f(x_0)}{2!h^2}(x - x_0)(x - x_1) +$$
$$+ \ldots + \frac{\Delta^n f(x_0)}{n!h^n}(x - x_0)(x - x_1)\ldots(x - x_{n-1}).$$

## Індивідуальне завдання

2-й варіант

| $x$ | 0,101 | 0,106 | 0,111 | 0,116 | 0,121 | 0,126 | 0,131 | 0,136 | 0,141 | 0,146 |
|---|---|---|---|---|---|---|---|---|---|---|
| $y$ | 1,261 | 1,276 | 1,291 | 1,306 | 1,321 | 1,336 | 1,352 | 1,367 | 1,383 | 1,399 |

$x_0 = 0{,}1102$

## Хід роботи

**Код програми:**

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <vector>

#define ArraySize 10
using namespace std;

int factorial(int n) {
	if(!n)
		return 1;
	return factorial(n - 1) * n;
}

void Lagrange(vector<double> &X, vector<double> &Y, double x) {
	cout << "[ Polinom Lagrange ]" << endl;

	double result = 0;
	for(int i = 0; i < X.size(); i++) {

		double numerator = 1; // чисельник
		double denominator = 1; // знаменник

		for(int j = 0; j < X.size(); j++) {
			if(i != j) {
				numerator *= x - X[j];
				denominator *= X[i] - X[j];
			}
		}
```

```
                result += (numerator / denominator) * Y[i];

        }
        cout << endl << "[ " << setprecision(6) << "Result: " << result << " ]" << endl;
}

void Newton(vector<double> &X, vector<double> &Y, double X0) {
        cout << endl << endl << "================================[ Polinom Newthon
]================================" << endl;
        cout << " d1_f(x)  d2_f(x)  d3_f(x)  d4_f(x)  d5_f(x)  d6_f(x)  d7_f(x)  d8_f(x)
d9_f(x)" << endl;

        double result = Y[0];
        double h = X[1] - X[0];
        double DY[ArraySize][ArraySize] = { 0 };

        //Заповнення таблиці скінченних різниць
        for(int i = 0; i < ArraySize; i++) {
                DY[i][0] = Y[i];
        }

        for(int i = 1; i < ArraySize; i++) {
                for(int j = 0; j < ArraySize - 1; j++) {
                        if(j < ArraySize - i) {
                                DY[j][i] = DY[j + 1][i - 1] - DY[j][i - 1];
                                cout << showpos << fixed << setprecision(4) << setw(8) <<
DY[j][i] << " " << noshowpos;
                        } else {
                                cout << "   ---    ";
                        }
                }
                cout << endl;
        }
        //таблиця заповнена та виведена


        for(int i = 1; i < ArraySize; i++) {
                double numerator = DY[0][i];
                double denominator = factorial(i);

                for(int j = 0; j < i; j++) {
                        numerator *= X0 - X[j];
                        denominator *= h;
                }
                result += numerator / denominator;

        }
        cout << endl << "================================[ " << setprecision(5) << "Result:
" << result << " ]================================" << endl;
}

int main() {
        vector<double> X = { 0.101, 0.106, 0.111, 0.116, 0.121, 0.126, 0.131, 0.136, 0.141,
0.146 };
        vector<double> Y = { 1.261, 1.276, 1.291, 1.306, 1.321, 1.336, 1.352, 1.367, 1.383,
1.399 };
```

```
	cout << "==============================[ Tabular function V2 ]==============================\n" <<
			"X | 0.101 | 0.106 | 0.111 | 0.116 | 0.121 | 0.126 | 0.131 | 0.136 | 0.141 | 0.146 |\n" <<
			"Y | 1.261 | 1.276 | 1.291 | 1.306 | 1.321 | 1.336 | 1.352 | 1.367 | 1.383 | 1.399 |\n" <<
			"=================================================================================" << endl << endl;

	double X0 = 0.1102;
	Lagrange(X, Y, X0);
	Newton(X, Y, X0);
	return 0;
}
```

**Результат:**

```
Microsoft Visual Studio Debug Console
==============================[ Tabular function V2 ]==============================
X | 0.101 | 0.106 | 0.111 | 0.116 | 0.121 | 0.126 | 0.131 | 0.136 | 0.141 | 0.146 |
Y | 1.261 | 1.276 | 1.291 | 1.306 | 1.321 | 1.336 | 1.352 | 1.367 | 1.383 | 1.399 |
===================================================================================

[ Polinom Lagrange ]

[ Result: 1.28868 ]


================================[ Polinom Newthon ]================================
 d1_f(x)  d2_f(x)  d3_f(x)  d4_f(x)  d5_f(x)  d6_f(x)  d7_f(x)  d8_f(x)  d9_f(x)
 +0.0150  +0.0150  +0.0150  +0.0150  +0.0150  +0.0160  +0.0150  +0.0160  +0.0160
 -0.0000  +0.0000  -0.0000  +0.0000  +0.0010  -0.0010  +0.0010  +0.0000    ---
 +0.0000  -0.0000  +0.0000  +0.0010  -0.0020  +0.0020  -0.0010    ---      ---
 -0.0000  +0.0000  +0.0010  -0.0030  +0.0040  -0.0030    ---      ---      ---
 +0.0000  +0.0010  -0.0040  +0.0070  -0.0070    ---      ---      ---      ---
 +0.0010  -0.0050  +0.0110  -0.0140    ---      ---      ---      ---      ---
 -0.0060  +0.0160  -0.0250    ---      ---      ---      ---      ---      ---
 +0.0220  -0.0410    ---      ---      ---      ---      ---      ---      ---
 -0.0630    ---      ---      ---      ---      ---      ---      ---      ---

================================[ Result: 1.28868 ]================================

D:\University\2_semester\Numerical Methods\Lab 8\x64\Debug\Lab 8.exe (process 7468) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when de
bugging stops.
Press any key to close this window . . .
```

# Висновок

Виконуючи лабораторну роботу, я ознайомився з методом інтерполяції таблично заданих функцій. Та навчився програмувати поліноми Лагранжа та Ньютона.