Лабарторна робота №1

MAIN.CPP

#include <iostream>

#include <cmath>

using namespace std;

void printMatrix(int\*\* mat, int m)

{

for (int i = 0; i < m; i++)//Обходимо масив

{

for (int j = 0; j < m; j++)

cout << mat[i][j] << " ";//Виводимо значення

cout << endl;

}

}

int norma1(int\*\* mat, int m)

{

int max = -100;

int\* res = new int[m];

for (int i = 0; i < m; i++)//Знаходимо кількість рядків

res[i] = 0;

for (int i = 0; i < m; i++)

for (int j = 0; j < m; j++)

res[i] += mat[i][j];//Обраховуємо суму елементів рядка

for (int i = 0; i < m; i++)

if (res[i] > max)//Знаходимо рядок в якого сума чисел найбільша

max = res[i];

return max;

}

int norma2(int\*\* mat, int m)

{

int max = -100;

int\* res = new int[m];

for (int i = 0; i < m; i++)//Знаходимо кількість стовпчиків

res[i] = 0;

for (int i = 0; i < m; i++)

for (int j = 0; j < m; j++)

res[i] += mat[j][i];//Обраховуємо суму елементів стовпчика

for (int i = 0; i < m; i++)

if (res[i] > max)//Знаходимо стовпчик в якого сума чисел найбільша

max = res[i];

return max;

}

double norma3(int\*\* mat, int m)

{

double res = 0.0;

for (int i = 0; i < m; i++)

for (int j = 0; j < m; j++)

res += pow(mat[i][j], 2);//Підносимо кожен елемент матриці до квадрату і знаходимо їх суму

res = pow(res, 0.5);//Знаходимо корінь цієї суми

return res;

}

int main(int argc, char const\* argv[])

{

int\*\* M;

int q;

cout << "Enter matrix: ";//Вводимо розміри матриці

cin >> q;

M = new int\* [q];//Cтворюємо масив

for (int i = 0; i < q; ++i)

{

M[i] = new int[q];

for (int j = 0; j < q; ++j)

{

cout << "A[" << i << "][" << j << "] = ";

cin >> M[i][j];//Заповнюємо масив числами

}

}

printMatrix(M, q);//Виводимо матрицю

cout << "Infinity norma = " << norma1(M, q) << endl;//Знаходимо нескінченну норму

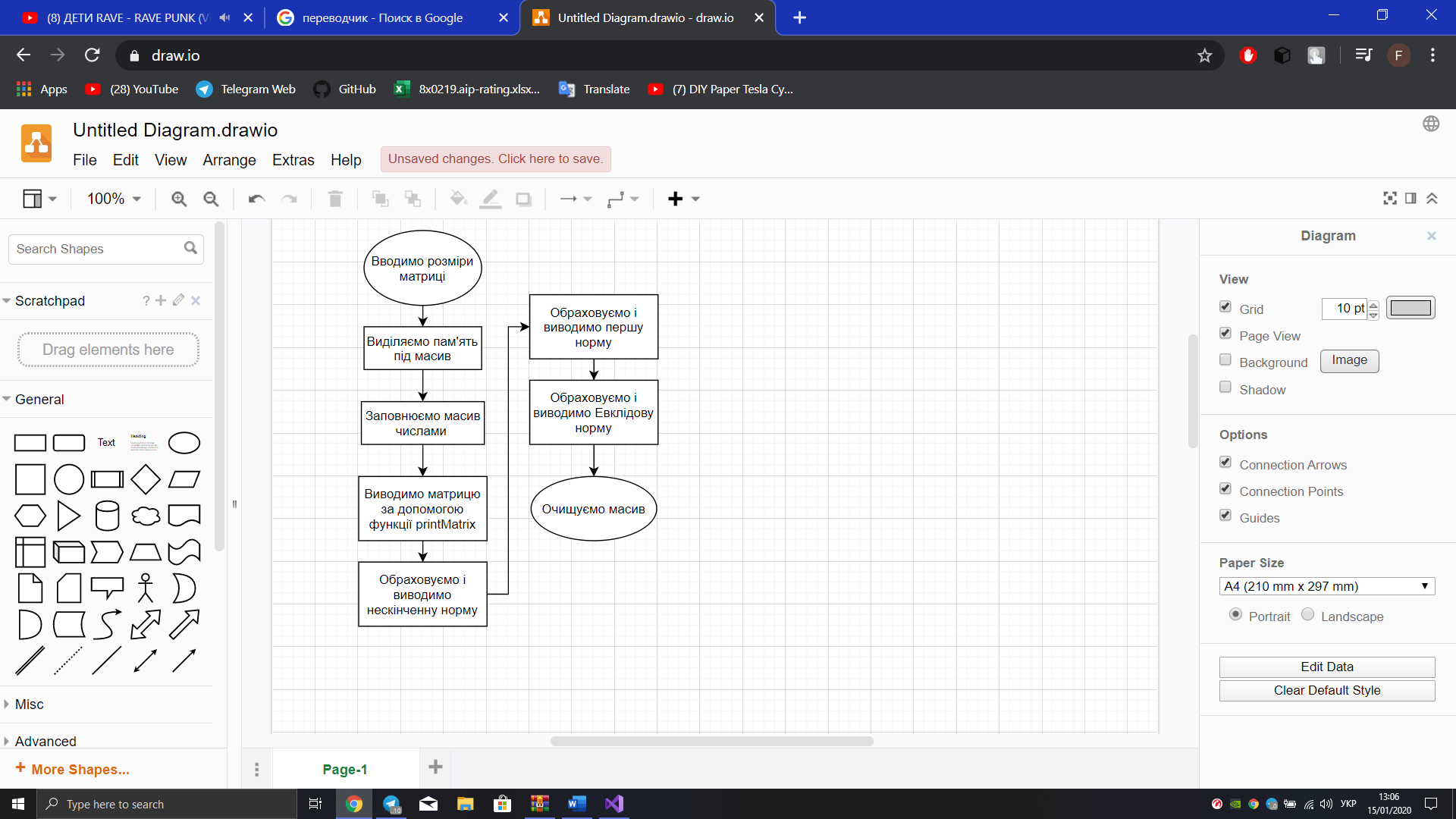
cout << "First norma = " << norma2(M, q) << endl;//Знаходимо першу норму

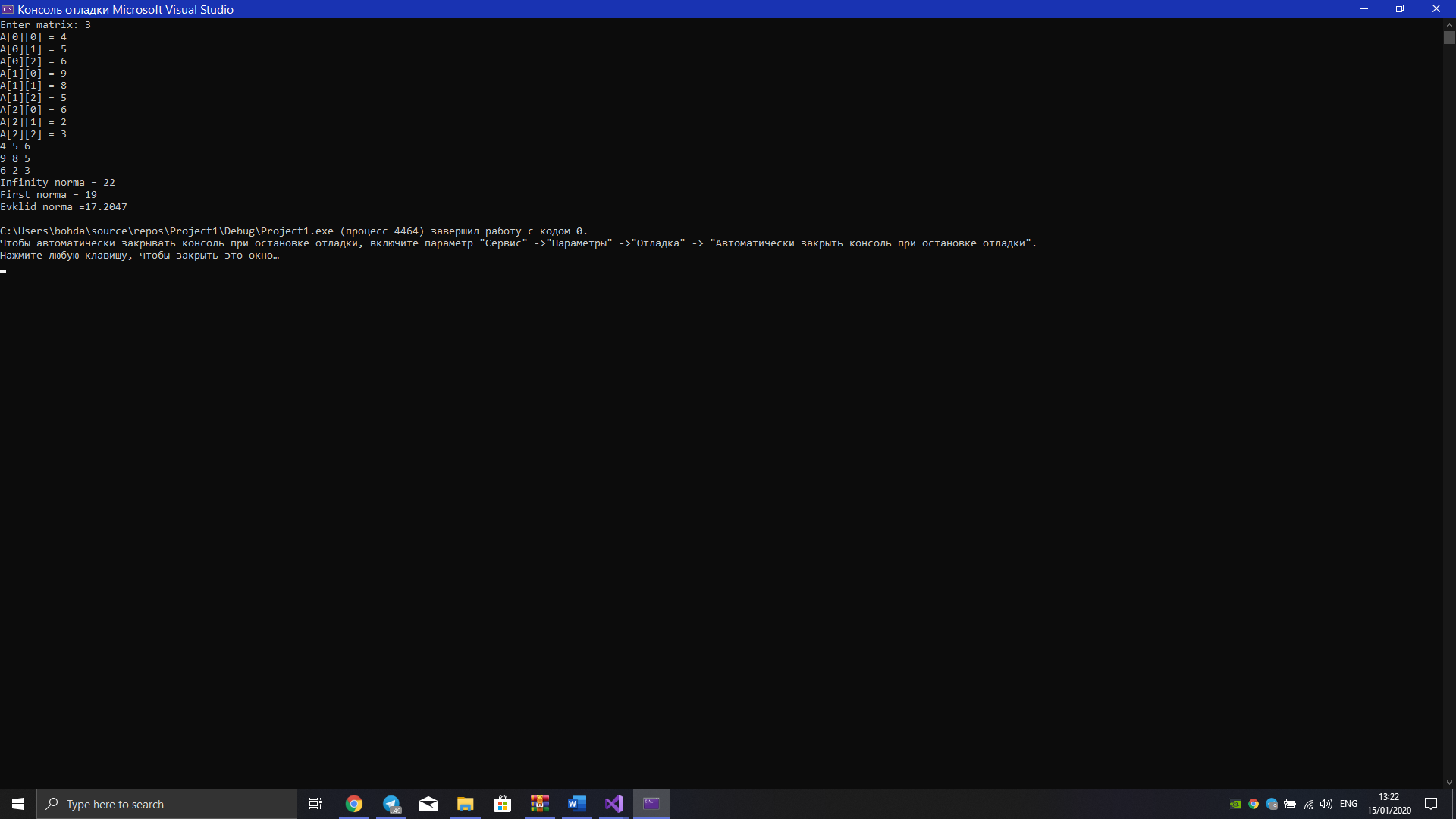
cout << "Evklid norma =" << norma3(M, q) << endl;//Знаходимо евклідову норму

return 0;

}

План





Лабарторна робота №2

MAIN.CPP

#include <iostream>

#include <math.h>

#include "header.h"

#include <glut.h>

using namespace std;

const int ScreenW = 1920;

const int ScreenH = 1080;

double scale = 3.0;

typedef struct tagpoint

{

double Ix;

double Iy;

double Lib;

}point;

int main(int argc, char\* argv[])

{

struct tagpoint point;

int n = 1;

int size = 100000;

int\* sizez = &size;

double x = 0.0;

double E = 0.001;

cout << "Enter x: ";

cin >> x;//Вводимо абсцису

x = checkx(x);//Перевіряємо чи х підходить умовам

double arctglib = atan (x);//Обраховуємо за допомогою математичної функції комп'ютерне значення х

cout << "Vikoristovuuchi bibliotechnu functciu = " << arctglib << endl;

double\* arr = new double[size];

row(arr, sizez, n, x, E);//Заповнюємо масив

printArr(arr, size);//Виводимо масив

point.Ix = x;

point.Iy = sumaInMatrix(arr, size);//Обраховуємо і записуємо значення ординати

point.Lib = arctglib;

wreadXY(point.Ix, 'x', 1, 0);

wreadXY(point.Iy, 'y', 1, 0);

wreadXY(point.Lib, 'Y', 1, 0);

delete[] arr;

glutInit(&argc, argv);

glutCreateWindow("Arctg");//Створення вікна із заданим ім'ям

glutInitWindowSize(ScreenW, ScreenH);//Ширина та висота вікна

glutInitWindowPosition(0, 0);//Положення вікна

glutKeyboardFunc(keyboard);//Підключаємо клавіатуру

glutDisplayFunc(display);

glutMainLoop();

return(0);

}

void display()

{

glMatrixMode(GL\_PROJECTION);//Тип матриці (проекційна)

glLoadIdentity();//Загрузка одиничної матриці

gluOrtho2D(0, ScreenW, 1, ScreenH);

double xd = 0.0;

double yd = 0.0;

double yL = 0.0;

xd = wreadXY(1, 'x', 0, 1);//Зчитуэмо значення точок

yd = wreadXY(1, 'y', 0, 1);

yL = wreadXY(1, 'Y', 0, 1);

double abpohibka = fabs(yL - yd);//Знаходимо абсолютну похибку

double vdpohibka = abpohibka/yd;//Знаходимо відносну похибку

cout << "Absolute pohibka: " << abpohibka << endl;

cout << "Vidnocna pohibka: " << vdpohibka << endl;

cout << "" << endl;

cout << "X: " << xd << endl;

cout << "Y: " << yd << endl;

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT);

double xscale = ScreenW / scale;

double yscale = ScreenH / scale;

glBegin(GL\_LINE\_STRIP);//Графік ф-ції

glColor3f(0.0f, 0.0f, 1.0f);

for (float x = -1; x <= 1; x += 0.0001)

{

float y = rowDisplay(x);

glVertex2f(x \* xscale + ScreenW / 2, y \* yscale + ScreenH / 2);

}

glEnd();

glFlush();

glBegin(GL\_LINES);//Вісь x на y

glColor3f(0.0f, 1.0f, 0.0f);

glVertex2f(-ScreenW, ScreenH / 2);

glVertex2f(ScreenW, ScreenH / 2);

glVertex2f(ScreenW / 2, -ScreenH);

glVertex2f(ScreenW / 2, ScreenH);

glEnd();

glFlush();

glPointSize(9);

glBegin(GL\_POINTS);

glColor3f(0.0f, 1.0f, 0.0f);//Колір точки

glVertex2f(xd \* xscale + ScreenW / 2, yd \* yscale + ScreenH / 2);

glEnd();

glFlush();

}

Лабарторна робота №3

MAIN.CPP

#include "tool.h"

#include <stdlib.h>

#include <glut.h>

#include <iostream>

#include <math.h>

using namespace std;

const int windowW = 600;

const int windowH = 600;

double scale = 1.0;

double step = 0.0;

static double xPoint;

static double yPoint;

int main(int argc, char\* argv[])

{

int size = 10;

float h = 0.1;

cout << "Enter x(0):";

float xNull = 0;

cin >> xNull;

cout << "Enter n:";

int n;

cin >> n;

float\* arrX = getArrX(xNull, h, size);

printArrX(arrX, size);

float\* arrY = getArrY(arrX, size);

float\*\* raznitsaArr = tabRaznitsa(arrY, size, n); //Розрахунок горизонтальної таблиці різниць

printArrInArr(raznitsaArr, size, n); //Cout горизонтальної таблиці різниць

cout << "\n Enter x which you want to search :" << endl << "x:";

float x;

cin >> x;//Вводимо x

float correctAtan = atan (x);

float q = 0;

q = (x - xNull) / h;

cout << "q = " << q << endl;

cout << "My interpolation = " << result(arrY, raznitsaArr, q, n) << endl;//Інтерполяція

yPoint = result(arrY, raznitsaArr, q, n);

xPoint = x;

cout << "Lib interpolation = " << atan(x) << endl;//Бібліотечна інтерполяція

cout << "Absolute mistage = " << fabs(correctAtan - result(arrY, raznitsaArr, q, n)) << endl;//Похибка

cout << "Enter scale:";

cin >> scale;

glutInit(&argc, argv);

glutCreateWindow("Atan"); //Створення вікна із заданим ім'ям

glutInitWindowSize(windowW, windowH); //Ширина та висота вікна

glutInitWindowPosition(0, 0); //Положення вікна

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

void display()

{

glMatrixMode(GL\_PROJECTION);//Тип матриці (проекційна)

glLoadIdentity();//Загрузка одиничної матриці

gluOrtho2D(-windowW, windowW, -windowH, windowH);//glOrto(left, right, bottom, top, zNear, zFar); gluOrto2D(left, right, bottom, top);

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT); //Очищення буферу кольору

float xscale = windowW / scale;

float yscale = windowH / scale;

glBegin(GL\_LINE\_STRIP); //Графік ф-ції

glColor3f(1.0f, 0.0f, 0.0f);

for (float x = -600; x <= 600; x += 0.001)

{

float y = atan(x);

glVertex2f(x \* xscale + step, y \* yscale + step);

}

glEnd();

glFlush();

glBegin(GL\_LINES);//Вісь x на y

glColor3f(0.0f, 0.0f, 1.0f);

glVertex2f(-windowW, 0);

glVertex2f(windowW, 0);

glVertex2f(0, -windowH);

glVertex2f(0, windowH);

glEnd();

glFlush();

glPointSize(4);

glBegin(GL\_POINTS);

glColor3f(0.0f, 1.0f, 0.0f);//Колір точки

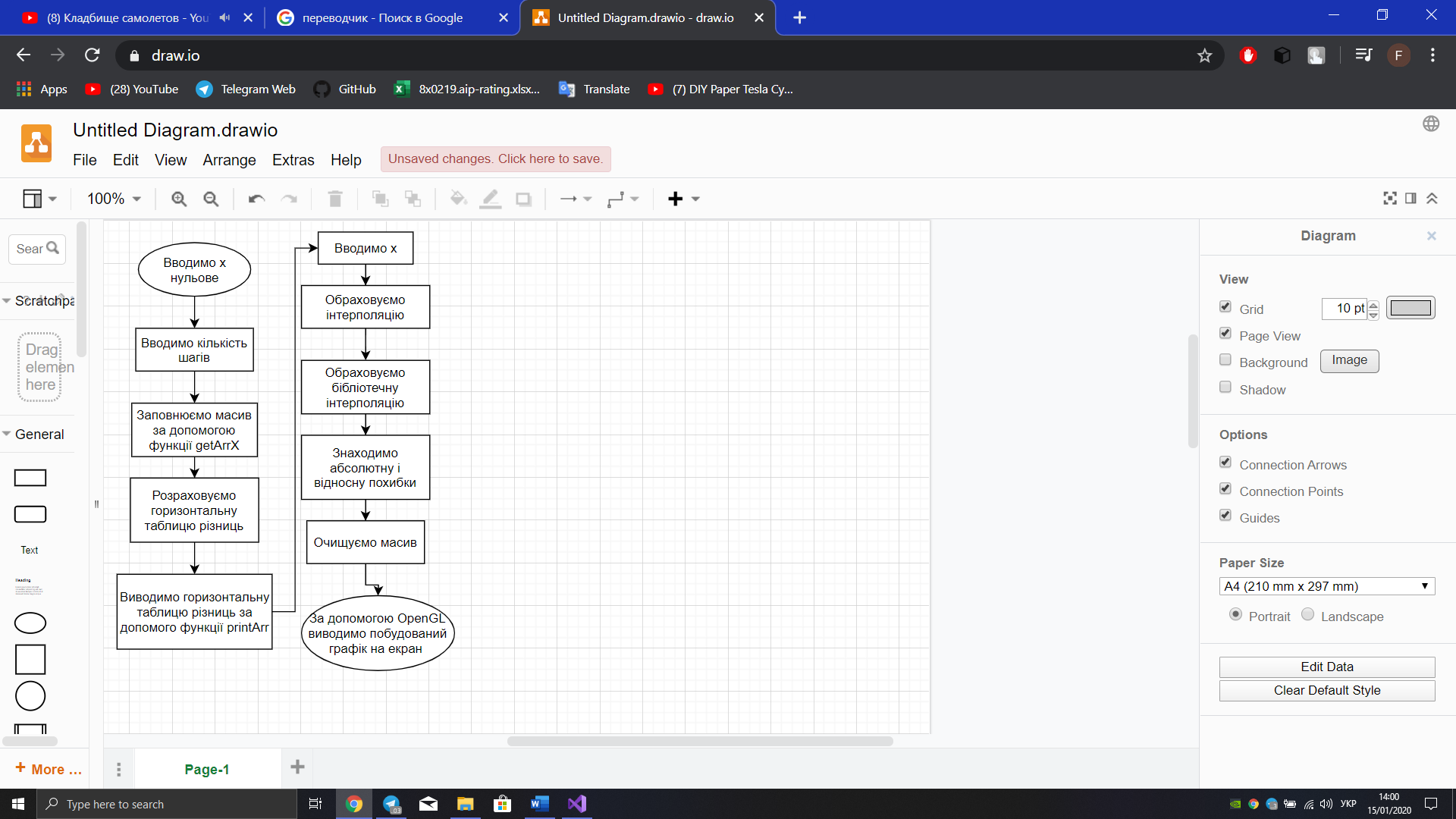
glVertex2f(xPoint \* xscale, yPoint \* yscale);

glEnd();

glFlush();

}

План



Лабарторна робота №5

MAIN.CPP

#include <iostream>

#include <math.h>

#define n 3

using namespace std;

double epsilon(double, double); //Відносна похибка

double delta(double, double); //Абсолютна похибка

int main(void)

{

setlocale(LC\_ALL, "Russian");

double xtrue, b, c, x[n], t[n] = { -0.707107, 0, 0.707107 }, res = 0; //Виділеня пам'яті і запис коефіцієнтів

cout << "Введите пределы интегрирования b и c: " << endl; //Ввод межей інтегрування

cout << "Нижний предел b = ";

cin >> b;

cout << "Верхний предел c = ";

cin >> c;

xtrue = 0.4388245731174946;//Точне значення

cout << "Комп'ютерное значение: " << xtrue << endl;

for (int i = 0; i < n; i++) //Обрахування точок х

{

x[i] = ((b + c) / 2) + ((c - b) \* t[i]) / 2;

}

for (int i = 0; i < n; i++) // Квадратурна формула Чебишева, 3 точки

{

double temp = (c - b) / n \* atan(x[i]);

res += temp;

}

cout << "Интеграл Чебышева: " << res << endl; // Вивід похибки

cout << "Абсолютная погрешность: " << delta(xtrue, res) << endl; // Вивід абсолютної похибки

cout << "Относительная погрешность: " << epsilon(xtrue, res) \* 100 << "%" << endl; // Вивід відносної похибки

system("pause");

return 0;

}

double epsilon(double xtrue, double xfalse)// Відносна похибка

{

double epsilon;

if (xfalse != 0)

{

epsilon = abs(xtrue - xfalse) / xtrue;

}

else epsilon = 0;

return epsilon;

}

double delta(double xtrue, double xfalse)//Абсолютна похибка

{

double delta = abs(xtrue - xfalse);

return delta;

}

План

