#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable : 4996)

#include <SFML/Graphics.hpp>

#include <iostream>

#include <windows.h>

#include <sstream>

#include <functional>

#include <fstream>

#include <iomanip>

#include <cmath>

#include <map>

#include <string>

using namespace std;

using namespace sf;

//Функции

//Конвертер в строку

template<typename T>

string convert\_to\_string(const T&);

//Облако точек функции

void function\_point\_cloud

(

const function<double(double)>& f,

vector<pair<double, double>>& v,

const double& min,

const double& max,

const double& step

);

//Отрисовка графика

Image graph\_builder

(

const vector<vector<pair<double, double>>>& Data\_,

const double& limit\_x0,

const double& limit\_x1,

const double& limit\_y0,

const double& limit\_y1,

const double& step\_x,

const double& step\_y,

const vector<Color>& colors,

const string& type,

const string& win\_name,

const size\_t& sz\_win\_x,

const size\_t& sz\_win\_y,

const size\_t& margin,

const size\_t& marker\_size,

const int& shift\_of\_axis\_x,

const int& shift\_of\_axis\_y,

const size\_t& thickness\_axis,

const string& x\_name,

const string& y\_name,

const Color& color\_axis,

const Color& color\_auxiliary,

const size\_t& thickness\_auxiliary\_line,

const size\_t& font\_size

);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("color 0A");

//+++++Графики

//Обёртка для функции

function<double(double)> f;

//Обёртка для параметрической функции

function<double(double)> fp;

//Контейнеры

//Пакет функций

vector<vector<pair<double, double>>> funcs;

//Пакет точек функции

vector<pair<double, double>> Data\_;

//Конфигурация системы отрисовки графиков

double x\_min = -12,

x\_max = 12,

step = 0.05,

min\_y = -12,

max\_y = 12;

//Палитра цветов линий графиков для функций в порядке следования их наборов точек в контейнере

vector<Color> colors;

colors.push\_back(Color(0, 255, 0));

colors.push\_back(Color(255, 0, 0));

colors.push\_back(Color(255, 255, 0));

colors.push\_back(Color(150, 75, 0));

colors.push\_back(Color(225, 165, 0));

colors.push\_back(Color(66, 170, 255));

//Задача функций

//Функция 1

f = [](double x)

{

double y = 2\*x-8;

return y;

};

function\_point\_cloud(f, Data\_, x\_min, x\_max, step);

funcs.push\_back(Data\_);

//Функция 2

Data\_.clear();

fp = [](double x)

{

double y = (5 - 3\*x)/2;

return y;

};

function\_point\_cloud(fp, Data\_, x\_min, x\_max, step);

funcs.push\_back(Data\_);

//Отрисовка пакета графиков

auto img\_graph = graph\_builder

(

funcs,

x\_min,

x\_max,

min\_y,

max\_y,

1.,

1.,

colors,

"line",

"Graphic",

1020u,

1020u,

20u,

3u,

0,

0,

3u,

"x",

"y",

Color::White,

Color::Cyan,

1u,

18u

);

img\_graph.saveToFile("Graph.jpg");

system("pause");

return 0;

}

//+++++Функции

//Конвертер в строку

template<typename T>

string convert\_to\_string(const T& value)

{

stringstream ss;

string output;

ss << value;

ss >> output;

return output;

}

//Облако точек функции

void function\_point\_cloud

(

const function<double(double)>& f,

vector<pair<double, double>>& v,

const double& min,

const double& max,

const double& step

)

{

for (double u = min; u <= max; u += step)

{

v.push\_back(make\_pair(u, f(u)));

}

}

//Отрисовка графика

static Image graph\_builder

(

const vector<vector<pair<double, double>>>& Data\_,

const double& limit\_x0,

const double& limit\_x1,

const double& limit\_y0,

const double& limit\_y1,

const double& step\_x,

const double& step\_y,

const vector<Color>& colors,

const string& type,

const string& win\_name,

const size\_t& sz\_win\_x,

const size\_t& sz\_win\_y,

const size\_t& margin,

const size\_t& marker\_size,

const int& shift\_of\_axis\_x,

const int& shift\_of\_axis\_y,

const size\_t& thickness\_axis,

const string& x\_name,

const string& y\_name,

const Color& color\_axis,

const Color& color\_auxiliary,

const size\_t& thickness\_auxiliary\_line,

const size\_t& font\_size

)

{

//Число Пи

double pi = atan(1.) \* 4.;

//Скриншот

Image img;

//Смещение центра системы координат

int shift\_of\_center\_x = static\_cast<double>(sz\_win\_x) / 2.;

int shift\_of\_center\_y = static\_cast<double>(sz\_win\_y) / 2.;

//Толщина линий графиков

size\_t thickness\_gr = marker\_size;

//Цвет графика

Color color\_graph;

RenderWindow window(VideoMode(sz\_win\_x, sz\_win\_y), win\_name);

cout << "Чтобы программа продолжила выполнение закройте после просмотра окно графика (которое появится)" << endl;

cout << endl;

//Работа в окне

while (window.isOpen())

{

//Событие

Event event;

while (window.pollEvent(event))

{

if (event.type == Event::Closed)

window.close();

}

window.clear(Color::Black);

unsigned NULL\_x = 40;

int st\_y = 805;

//Загрузка шрифтов

sf::Font font;

if (!font.loadFromFile("C:\\Windows\\Fonts\\arial.ttf"))

{

cout << "Ошибка чтения шрифта" << endl;

}

Text text;

text.setFont(font);

text.setCharacterSize(font\_size); //В пикселях, а не точках!

text.setFillColor(sf::Color::Green);

text.setStyle(sf::Text::Bold);

//Прочерчивание осей

//Прямоугольник представляющий оси

RectangleShape rectangle0;

rectangle0.setFillColor(color\_axis);

//Вертикаль Y

rectangle0.setPosition(shift\_of\_center\_x + shift\_of\_axis\_y, margin);

rectangle0.setSize(Vector2f(thickness\_axis, sz\_win\_y - 2 \* margin));

window.draw(rectangle0);

//Горизонталь X

rectangle0.setPosition(margin, shift\_of\_center\_y + shift\_of\_axis\_x);

rectangle0.setSize(Vector2f(sz\_win\_x - 2 \* margin, thickness\_axis));

window.draw(rectangle0);

//Разметка вертикальной оси

unsigned count\_marker\_y = static\_cast<double>(limit\_y1 - limit\_y0) / step\_y;

unsigned len\_axis\_y = sz\_win\_y - 2 \* margin;

for (unsigned u = 0; u <= count\_marker\_y; ++u)

{

//Текст

text.setString(convert\_to\_string(limit\_y0 + (count\_marker\_y - u) \* step\_y));

text.setPosition(shift\_of\_center\_x + shift\_of\_axis\_y + 15, (static\_cast<double>(len\_axis\_y) / count\_marker\_y) \* u + margin - 10);

window.draw(text);

//Вспомогательные линии сетки

//Прямоугольник представляющий вспомогательные линии сетки

rectangle0.setFillColor(color\_auxiliary);

//Вертикальные

rectangle0.setPosition(shift\_of\_center\_x - margin + 15 + shift\_of\_axis\_y, (static\_cast<double>(len\_axis\_y) / count\_marker\_y) \* u + margin);

rectangle0.setSize(Vector2f(11, thickness\_auxiliary\_line \* 2));

window.draw(rectangle0);

//Горизонтальные

rectangle0.setPosition(margin, (static\_cast<double>(len\_axis\_y) / count\_marker\_y) \* u + margin);

rectangle0.setSize(Vector2f(sz\_win\_x - 2 \* margin, thickness\_auxiliary\_line));

window.draw(rectangle0);

}

//Разметка горизонатльной оси

unsigned count\_marker\_x = static\_cast<double>(limit\_x1 - limit\_x0) / step\_x;

unsigned len\_axis\_x = sz\_win\_x - 2 \* margin;

for (unsigned u = 0; u <= count\_marker\_x; ++u)

{

//Текст

text.setString(convert\_to\_string(limit\_x0 + u \* step\_x));

text.setPosition((static\_cast<double>(len\_axis\_x) / count\_marker\_x) \* u + margin, shift\_of\_center\_y + shift\_of\_axis\_x + 15);

window.draw(text);

//Вспомогательные линии сетки

rectangle0.setPosition((static\_cast<double>(len\_axis\_x) / count\_marker\_x) \* u + margin, shift\_of\_center\_y - margin + 15 + shift\_of\_axis\_x);

rectangle0.setSize(Vector2f(thickness\_auxiliary\_line \* 2, 11));

window.draw(rectangle0);

rectangle0.setPosition((static\_cast<double>(len\_axis\_x) / count\_marker\_x) \* u + margin, margin);

rectangle0.setSize(Vector2f(thickness\_auxiliary\_line, sz\_win\_y - 2 \* margin));

window.draw(rectangle0);

}

//Подпись имён осей

//X

text.setString(x\_name);

text.setPosition(sz\_win\_x - margin - 15, shift\_of\_center\_y + shift\_of\_axis\_x - 25);

window.draw(text);

//Y

text.setString(y\_name);

text.setPosition(shift\_of\_center\_x + shift\_of\_axis\_y - 15, margin);

window.draw(text);

//Отрисовка графика

//Коэффициенты масштабирования по осям

double scaling\_factor\_x = static\_cast<double>(sz\_win\_x - 2 \* margin) / (limit\_x1 - limit\_x0); //Пиксел в натуре

double scaling\_factor\_y = static\_cast<double>(sz\_win\_y - 2 \* margin) / (limit\_y1 - limit\_y0);

auto colors0 = colors;

for (auto g : Data\_)

{

color\_graph = colors0[0];

colors0.erase(colors0.begin());

pair<double, double> old;

bool trigger = true;

for (auto pr : g)

{

if (trigger)

{

trigger = false;

}

else

{

double point\_x\_old = (old.first - limit\_x0) \* scaling\_factor\_x + margin;

double point\_y\_old = sz\_win\_y + (limit\_y0 - old.second) \* scaling\_factor\_y - margin;

if (type == "line")

{

double point\_x\_next = (pr.first - limit\_x0) \* scaling\_factor\_x + margin;

double point\_y\_next = sz\_win\_y + (limit\_y0 - pr.second) \* scaling\_factor\_y - margin;

double len = sqrt(pow((point\_x\_next - point\_x\_old), 2) + pow((point\_y\_next - point\_y\_old), 2));

sf::RectangleShape line(sf::Vector2f(len, thickness\_gr));

line.setPosition(point\_x\_old, point\_y\_old);

double ang = (asin((point\_y\_next - point\_y\_old) / len) / pi) \* 180;

line.rotate(ang);

line.setFillColor(color\_graph);

window.draw(line);

}

else if (type == "points")

{

sf::CircleShape shape(marker\_size);

shape.setPosition(point\_x\_old - marker\_size, point\_y\_old - marker\_size);

shape.setFillColor(color\_graph);

window.draw(shape);

}

}

old = pr;

}

}

//Скрин графика

while (window.pollEvent(event))

{

switch (event.type)

{

case sf::Event::Closed:

{

img = window.capture();

window.close();

break;

}

default:

break;

}

}

//Отображение графика в окне

window.display();

}

cout << "График успешно построен" << endl;

cout << endl;

return img;

}