//

// Created on 06/09/2017.

//

// You are given a graphic application using FLTK 1.3.4 (â€‹http://www.fltk.org)

// There are some errors, which were made specifically for you. Please do not change the main() function,

// find and fix all the errors to get the program compiled and linked. after that, please run and check

// the result, it must look similar to TestTask.png"

// Please send your corrected code with a build instruction and a CV in one zip file to training@coccoc.com

//

// Enjoy and good luck!

//

#include <iostream>

#include <FL/Fl.H>

#include <FL/fl\_draw.H>

#include <FL/Fl\_Window.H>

#include <initializer\_list>

#include <vector>

#include <functional>

#include <math.h>

struct Point {

int x,y;

Point(int xx, int yy) : x(xx), y(yy) { }

};

class Shape {

public:

Point point(int idx) const {

return points[idx];

}

void draw() const {

draw\_lines();

}

void add(Point p){ points.push\_back(p); }

protected:

virtual void draw\_lines() const {

for (unsigned int i=1; i<points.size(); ++i)

fl\_line(points[i-1].x, points[i-1].y, points[i].x, points[i].y);

}

private:

std::vector<Point> points;

};

class ClosedPolyline: public Shape {

public:

ClosedPolyline(std::initializer\_list<Point> pp) {

if (pp.size() > 0) {

for (auto &p: pp)

add(p);

}

}

};

class MyWindow: public Fl\_Window {

public:

MyWindow(int x, int y, int w, int h, const char\* title = 0)

: Fl\_Window(x, y, w, h, title) {}

void Attach(Shape s) {

shapes.push\_back(&s);

}

protected:

void draw() override {

for (auto s: shapes) {

s->draw();

}

}

private:

std::vector<Shape\*> shapes;

};

class Circle: public Shape {

public:

Circle(Point p, double r): radius(r) {

add(Point{p.x - r, p.y - r});

}

protected:

void draw\_line() {

fl\_arc(point(0).x, point(0).y, radius + radius,radius + radius, 0, 360);

}

private:

double radius;

};

Function(T f, double r1, double r2,

Point xy, int count = 100, double xscale = 25, double yscale = 25) {

double dist = (r2-r1)/count;

double r = r1;

for (int i = 0; i<count; ++i) {

add(Point(xy.x+int(r\*xscale), xy.y-int(f(r)\*yscale)));

r += dist;

}

}

};

class CosFunction {

public:

double (double x) {

return cos(x \* M\_PI / 180);

}

};

typedef double Fct(double);

int main() {

MyWindow win(100, 100, 600, 400, "C++ Test task"); // Create a simple window

ClosedPolyline p{Point{100, 100}, Point{100, 200}, Point{500, 100}, Point{500, 200}};

Function<std::function<double(double)>> f1 {[] (double x) -> double { return x \* x; }, -100, 100, Point {300, 300}, 100, 20, 5};

Function<Fct> f2 {sin, -360, 360, Point{300, 300}, 200, 1, 25};

Function<CosFunction> f3{CosFunction(), -360, 360, Point{300, 300}, 200, 1, 25};

Circle c1{Point{300, 50}, 30};

win.Attach(p);

win.Attach(f1);

win.Attach(f2);

win.Attach(f3);

win.Attach(c1);

win.end();

win.show();

return (Fl::run());

}

