**Ćwiczenie 6 Rozdział 15**

Z jakiegoś powodu muszę mieć w każdej nowej klasie która dziedziczy po Shape funkcję

void draw\_lines() const;

Przynajmniej jeśli w tej klasie używasz obiektów innych klas niż Point np. Rectangle, Polygon czy Open\_polyline, bo funkcja add() dla Point działa bez draw\_lines().

Bez draw\_lines() dodając obiekt nowo utworzonej klasy do okna za pomocą win.attach(nowa\_klasa) nie będziesz widział utworzonej w niej figur takich jak Rectangle, Polygon czy Open\_polyline.

Najlepsze, że nigdzie ta funkcja nie jest bezpośrednio wywoływana nawet w konstruktorze klasy Shape :O Po postu bez tego nie działa. Poniżej jak mniej więcej to powinno wyglądać, aby działało.

**Plik .h**

struct Bar\_chart : public Graph\_lib::Shape

{

Bar\_chart(vector<double> values, Point orig, double width, double xscale, double yscale);

void draw\_lines() const;// !!!!!!!!!!!!!!!!!!!!!!!!!!! I have no idea why but the code does not work without it. I Think we need to inherit this function.

//Hovewer where is this function invoked?! I am not using it explicitly, have no idea what is happening....

Vector\_ref<Rectangle> val;

};

**Plik .cpp**

Bar\_chart::Bar\_chart(vector<double> values, Point orig, double width, double xscale, double yscale)

{

double bar\_width = 0;

//val.push\_back(new Rectangle(Point(orig.x, orig.y - 4 \* yscale), 1 \* xscale, 4 \* yscale));

for (int i = 0; i < values.size(); ++i)

{

val.push\_back(new Rectangle( Point(orig.x + bar\_width, orig.y - values[i] \* yscale), 1 \* xscale, values[i] \* yscale) );

bar\_width += width;

}

}

void Bar\_chart::draw\_lines() const

{

Shape::draw\_lines();

for (int i = 0; i < val.size(); ++i)

{

val[i].draw();

}

}

**Funkcja która idzie do main()**

int homework()

{

//const int MAX\_FCT = 21;

//compare\_factorials(MAX\_FCT); //exercise 1

const int xmax = 600;

const int ymax = 600;

const int xoffset = 100;

const int yoffset = 100;

const int xspace = 100;

const int yspace = 100;

const int xlength = xmax - xoffset - xspace; // 400 pixeles

const int ylength = ymax - yoffset - yspace; // 400 pixeles

const int xscale = 20;

const int yscale = 20;

const int r\_min = -10;

const int r\_max = 11;

const int number\_of\_notches = r\_max - r\_min -1; // 20

const int scaled\_length = abs(r\_min - r\_max) - 1; // length of 20 on axis which is eqivalent of 400 pixels

const Point orig(xmax / 2, ymax / 2); // Point(xmax/2, ymax/2) corresponds to Point(0,0) on displayed label

const int graph\_resolution = 300;

const double PI = \_Pi;

const double graph\_presicion = 1; // 1 is the best presicion, precision closer to one - better

Simple\_window win(Point(100, 100), 600, 600, "Homework chapter 15");

Axis x\_axis(Axis::x, Point(xoffset, ymax / 2), xlength, number\_of\_notches, "x asis");

Axis y\_axis(Axis::y, Point(xmax / 2, ymax - yoffset), ylength, number\_of\_notches, "y asis"); // Axis goes up from point (ymax-yoffset) = 500, to ylength = 100

win.attach(x\_axis);

win.attach(y\_axis);

vector<double> vals = { 9.2, 4,3,1,7,5 };

Bar\_chart bar\_ch(vals,orig,xscale,xscale,yscale);

win.attach(bar\_ch);

win.wait\_for\_button();

return 0;

}

}

**main()**

int main()

{

system("chcp1250");

try {

H15::homework();

/\*

Ex15::exercises();

Ex14::cwiczenia();

Ex13::cwiczenia();

Ex12::cwiczenia();

\*/

system("pause");

}

catch (exception& e) {

cerr << "exception: " << e.what() << endl;

keep\_window\_open();

}

catch (...) {

cerr << "exception\n";

keep\_window\_open();

}

return 0;

}