**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №2**

**Дисциплина: Компьютерная графика**

**Тема: Построение графиков функций**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: Осипов Олег Васильевич

Белгород 2019

**Цель работы:** разработка модуля для построения графиков функций одной переменной на декартовой плоскости в среде Qt Creator.

**Порядок выполнения работы**

1. Разработать визуальный компонент Chart для построения графиков функций одной переменной на декартовой плоскости.
2. Протестировать и отладить программу с различными функциями.

**Требования к программе**

1. В качестве входных данных пользователь вводит интервал по оси абсцисс , на котором строится график. Программа при этом определяет автоматически, в каких пределах изменяются значения *y*.
2. Реализовать функцию step для разметки осей.
3. Реализовать классы, описанные в разделе 3.
4. Предусмотреть возможность построения нескольких функций на одном графике.
5. Программа должна при изменении размера окна пересчитывать координаты разметок и перестраивать график.
6. Числовые подписи к разметкам (линиям сетки) должны быть выравнены строго по центру разметки (как на рис. 2 и рис. 3).
7. Ограничить область значений функций большими константами *y*max, *y*min (для построения, например, гиперболы вблизи *x*=0).

**Дополнительные задания**

1. Если начало координат (x=0; y=0) попадает в область видимости, то выводить оси и разметку из данной точки (по умолчанию оси можно рисовать из нижнего левого угла).
2. Реализовать ввод функций в символьном виде во время выполнения программы. Для этого необходимо изучить материал по динамической компиляции и рефлексии на C# (функцию eval для Java) или другом языке программирования.

Axis.h

#ifndef AXIS\_H

#define AXIS\_H

#include <QPainter>

#include <cmath>

#include "coords.h"

#define PREFER\_H 45

class Axis

{

public:

static void step(double &k, double &H, int &m, int &n);

};

class AbscissaAxis: public Axis

{

public:

void draw( ScreenCoords scr, WorldCoords w, QPainter &p);

};

class OrdinateAxis: public Axis

{

public:

void draw(ScreenCoords scr, WorldCoords w, QPainter &p);

};

Axis.cpp

#include "axis.h"

#include <QtCore>

void Axis::step(double &k, double &H, int &m, int &n)

{

double p = H/k;

double n1 = log10(p);

double n2 = log10(p/2);

double n5 = log10(p/5);

m = 1; n = int(floor(n1));

double min\_ = p - pow(10, floor(n1));

if (p - 2\*pow(10, floor(n2)) < min\_)

{

min\_ = p - 2\*pow(10, floor(n2));

m = 2; n = int(floor(n2));

}

if (p - 5\*pow(10, floor(n5)) < min\_)

{

min\_ = p - 5\*pow(10, floor(n5));

m = 5; n = int(floor(n5));

}

if (pow(10, ceil(n1)) - p < min\_)

{

min\_ = pow(10, ceil(n1)) - p;

m = 1; n = int(ceil(n1));

}

if (2\*pow(10, ceil(n2)) - p < min\_)

{

min\_ = 2\*pow(10, ceil(n2)) - p;

m = 2; n = int(ceil(n2));

}

if (5\*pow(10, ceil(n5)) - p < min\_)

{

min\_ = 5\*pow(10, ceil(n5)) - p;

m = 5; n = int(ceil(n5));

}

double h = m\*pow(10, n);

H = h\*k;

k = h;

}

void AbscissaAxis::draw(ScreenCoords s, WorldCoords w, QPainter &p)

{

double k = (s.xend - s.xbeg)/(w.xend - w.xbeg);

double H = PREFER\_H;

int m = 0, n = 0;

step(k, H, m, n);

QPointF a(s.xbeg, s.yend);

QPointF b(s.xend, s.yend);

p.setPen(QPen(Qt::black, 100));

p.setBrush(Qt::black);

p.drawLine(a, b);

double xWorld0 = (floor(w.xbeg/k) + 1)\*k;

while (xWorld0 <= w.xend)

{

QPointF serif = Converter::worldToScreen(QPointF(xWorld0, w.ybeg), w, s);

serif.setY(serif.y() - 10);

QPointF serifEnd = serif;

serifEnd.setY(serif.y() + 10);

p.drawLine(serif, serifEnd);

QString s = QString::number(xWorld0);

p.drawText(serifEnd.x() - 5, serifEnd.y() + 15, s);

xWorld0 += k;

}

}

void OrdinateAxis::draw(ScreenCoords s, WorldCoords w, QPainter &p)

{

double k = (s.yend - s.ybeg)/(w.yend - w.ybeg);

double H = PREFER\_H;

int m = 0, n = 0;

step(k, H, m, n);

QPointF a(s.xbeg, s.yend);

QPointF b(s.xbeg, s.ybeg);

p.setPen(QPen(Qt::green, 100));

p.setBrush(QBrush(Qt::green));

p.drawLine(a, b);

double yWorld0 = (floor(w.ybeg/k) + 1)\*k;

while (yWorld0 <= w.yend)

{

QPointF serif = Converter::worldToScreen(QPointF(w.xbeg, yWorld0), w, s);

serif.setX(serif.x() + 10);

QPointF serifEnd = serif;

serifEnd.setX(serif.x() - 10);

p.drawLine(serif, serifEnd);

QString s = QString::number(yWorld0);

p.drawText(serifEnd.x() - 20, serifEnd.y() + 5, s);

yWorld0 += k;

}

}

Chart.h

#ifndef CHART\_H

#define CHART\_H

#include <QWidget>

#include "math.h"

#include <QPen>

#include <QPainter>

using namespace std;

class Chart : public QWidget

{

Q\_OBJECT

public:

explicit Chart(QWidget \*parent = nullptr);

float globMAX=0;

float a=3.777,b=3.99,step=0.0001;

float offsetx=0, offsety=0;

class CurveData

{

QColor curveColor; // Цвет кривой

QPen curvePen; // Стиль пера

public:

virtual double F(double x) { return 0; }; // Функция для рисования

};

// Массив указателей на структуры, содержащие информацию о функциях f(x)

QVector<CurveData\*> curves;

// Функция устанавливает область определения графика функции

void setRangeX(double x1, double x2);

class SinCurveData : public CurveData

{

public:

double F(double x) override

{

return sin(x);

}

};

void redraw();

class WorldToScreenConverter

{

public:

// Экранная система координат

double X1, X2, Y1, Y2;

// Мировая система координат

double x1, x2, y1, y2;

// Функция преобразует координаты точки (x, y) из мировой системы координат в экранную

QPoint convert(double x, double y);

// Функция преобразует значение ординаты Y из экранной системы координат в мировую

double y(int Y);

// Функция возвращает значение абсциссы в мировой системе координат по её значению X в экранной

double x(int X);

};

WorldToScreenConverter coords\_converter;

private:

QPainter painter;

QRect srcwidgetsize, srcwindowsize;

// Объект для преобразования мировых координат в экранные и наоборот

void paintEvent(QPaintEvent\* p\_event);

void resizeEvent(QResizeEvent\* r\_event);

signals:

public slots:

};

#endif // CHART\_H

Chart.cpp

#include "chart.h"

#include "math.h"

#include <QApplication>

#include <QDesktopWidget>

#include "QPainter"

#include "axis.h"

#include "coords.h"

#include "mainwindow.h"

#include <QDebug>

using namespace std;

Chart::Chart(QWidget \*parent) : QWidget(parent)

{

}

void Chart::redraw()

{

}

void Chart::paintEvent(QPaintEvent \*p\_event)

{

this->resize(srcwidgetsize.width() \* this->parentWidget()->width()/srcwindowsize.width(),

srcwidgetsize.height() \* this->parentWidget()->height()/srcwindowsize.height());

this->Chart::coords\_converter.X1 = this->x();

this->Chart::coords\_converter.Y1 = this->y();

this->Chart::coords\_converter.X2 = this->rect().bottomRight().x()-30;

this->Chart::coords\_converter.Y2 = this->rect().bottomRight().y()-30;

this->Chart::coords\_converter.x1 = 0;

this->Chart::coords\_converter.y1 = 0;

this->Chart::coords\_converter.x2 = 10;

this->Chart::coords\_converter.y2 = 10;

AbscissaAxis absaxis;

OrdinateAxis ordaxis;

globMAX=0;

float globMIN=0;

for (float i=this->a;i<this->b;i+=0.1) {

if((curves[0]->F(i))>globMAX) globMAX = (curves[0]->F(i));

if((curves[0]->F(i))<globMIN) globMIN = (curves[0]->F(i));

}

qDebug()<<globMAX<<" "<<globMIN;

WorldCoords wc(a+offsetx,b+offsetx,globMIN,globMAX);

ScreenCoords sc(coords\_converter.X1,coords\_converter.X2,coords\_converter.Y1,coords\_converter.Y2);

vector<QPoint> points;

for (int i=0;i<curves.length();i++) {

for (float j=this->a; j<b; j+=0.0005){

QPoint monitorcoord = Converter::worldToScreen(QPointF(j,curves[i]->F(j)),wc,sc).toPoint();

points.push\_back(monitorcoord);

}

}

QPainter painter(this);

this->painter.begin(this);

painter.setRenderHint(QPainter::HighQualityAntialiasing);

QRect a = this->rect();

painter.setPen(Qt::black);

painter.setBrush(Qt::black);

painter.drawPoints(&points[0],points.size());

absaxis.draw(sc, wc, painter);

ordaxis.draw(sc, wc, painter);

this->painter.end();

redraw();

update();

}

void Chart::resizeEvent(QResizeEvent \*r\_event)

{

srcwidgetsize.setCoords(0,0,this->size().width(),this->size().height());

srcwindowsize.setCoords(0,0,this->parentWidget()->size().width(),this->parentWidget()->size().height());

}

Coords.h

#ifndef COORDS\_H

#define COORDS\_H

#include <QPoint>

class Coords

{

public:

double xbeg=0, xend=0, ybeg=0, yend=0;

//Coords(double xbeg\_, double xend\_, double ybeg\_, double yend\_);

QPointF topLeft();

QPointF topRight();

QPointF bottomLeft();

QPointF bottomRight();

double Xbeg();

double Xend();

double Ybeg();

double Yend();

virtual void setXbeg(double);

virtual void setXend(double);

virtual void setYbeg(double);

virtual void setYend(double);

virtual ~Coords(){}

};

class ScreenCoords: public Coords

{

public:

double xbeg=0, xend=0, ybeg=0, yend=0;

ScreenCoords(double xbeg\_, double xend\_, double ybeg\_, double yend\_)

{

xbeg = xbeg\_;

xend = xend\_;

ybeg = ybeg\_;

yend = yend\_;

}

QPointF topLeft();

QPointF topRight();

QPointF bottomLeft();

QPointF bottomRight();

};

//xbeg, ybeg matching to left bottom corner

class WorldCoords: public Coords

{

public:

double xbeg=0, xend=0, ybeg=0, yend=0;

WorldCoords(double xbeg\_, double xend\_, double ybeg\_, double yend\_)

{

xbeg = xbeg\_;

xend = xend\_;

ybeg = ybeg\_;

yend = yend\_;

}

QPointF topLeft();

QPointF topRight();

QPointF bottomLeft();

QPointF bottomRight();

};

class Converter

{

public:

static QPointF screenToWorld(QPointF scrPoint, ScreenCoords scr, WorldCoords world);

static QPointF worldToScreen(QPointF worldPoint, WorldCoords world, ScreenCoords scr);

static QPointF worldToWorld(QPointF w1Point, WorldCoords world1, WorldCoords world2);

static QPointF screenToScreen(QPointF s1Point, ScreenCoords screen1, ScreenCoords screen2);

};

#endif // COORDS\_H

Coords.cpp

#include "coords.h"

#include <QtCore>

double Coords::Xbeg()

{

return xbeg;

}

double Coords::Xend()

{

return xend;

}

double Coords::Ybeg()

{

return ybeg;

}

double Coords::Yend()

{

return yend;

}

void Coords::setXbeg(double xbeg\_)

{

xbeg = xbeg\_;

}

void Coords::setXend(double xend\_)

{

xend = xend\_;

}

void Coords::setYbeg(double ybeg\_)

{

ybeg = ybeg\_;

}

void Coords::setYend(double yend\_)

{

yend = yend\_;

}

QPointF Converter::screenToWorld(QPointF scrPoint, ScreenCoords scr, WorldCoords world)

{

QPointF res;

double coeff1 = (world.xend - world.xbeg)/(scr.xend - scr.xbeg);

double coeff2 = (world.yend - world.ybeg)/(scr.yend - scr.ybeg);

res.setX(world.xbeg + coeff1\*(scrPoint.x() - scr.xbeg));

res.setY(world.yend - coeff2\*(scrPoint.y() - scr.ybeg));

return res;

}

QPointF Converter::worldToScreen(QPointF worldPoint, WorldCoords world, ScreenCoords scr)

{

QPointF res;

double coeff1 = (scr.xend - scr.xbeg)/(world.xend - world.xbeg);

double coeff2 = (scr.yend - scr.ybeg)/(world.yend - world.ybeg);

res.setX(scr.xbeg + coeff1\*(worldPoint.x() - world.xbeg));

res.setY(scr.yend - coeff2\*(worldPoint.y() - world.ybeg));

return res;

}

Mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include "chart.h"

#include <QMainWindow>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = nullptr);

Ui::MainWindow \*ui;

~MainWindow();

public slots:

void showCOORDS();

private slots:

void on\_horizontalSlider1\_valueChanged(int value);

void on\_horizontalSlider\_2\_valueChanged(int value);

void on\_rangeEdit\_textEdited(const QString &arg1);

private:

QTimer \*timer;

Chart\* chartlink;

void resizeEvent(QResizeEvent\* r\_event);

};

#endif // MAINWINDOW\_H

Mainwindow.cpp

#include "chart.h"

#include "coords.h"

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QTimer>

#include <QDesktopWidget>

#include <QDebug>

class x3CurveData : public Chart::CurveData

{

public:

double F(double x) override

{

return 3\*x+2;

}

};

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

//ui->rangeEdit->text();

ui->mychar->curves.push\_back(new x3CurveData());

chartlink = ui->mychar;

//ui->mychar->;

timer = new QTimer(this);

QObject::connect(timer, SIGNAL(timeout()), this, SLOT(showCOORDS()));

timer->start(60); //time specified in ms

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::showCOORDS()

{

QPoint globalCursorPos = QCursor::pos();

int mouseScreen = qApp->desktop()->screenNumber(globalCursorPos);

QRect mouseScreenGeometry = qApp->desktop()->screen(mouseScreen)->geometry();

QPoint localCursorPos = globalCursorPos - mouseScreenGeometry.topLeft();

this->parentWidget()->mapToGlobal(this->pos());

QPointF mouse\_coord = QPoint(localCursorPos.x()-ui->mychar->mapToGlobal(ui->mychar->pos()).x(),((ui->mychar->mapToGlobal(ui->mychar->pos()).y())-localCursorPos.y())+(ui->mychar->coords\_converter.Y2-ui->mychar->coords\_converter.Y1));

WorldCoords wc(ui->mychar->coords\_converter.x1,ui->mychar->coords\_converter.x2,ui->mychar->coords\_converter.y1,ui->mychar->coords\_converter.y2);

ScreenCoords sc(0,(ui->mychar->coords\_converter.X2-ui->mychar->coords\_converter.X1),0,(ui->mychar->coords\_converter.Y2-ui->mychar->coords\_converter.Y1));

QPointF monitorcoord = Converter::screenToWorld(mouse\_coord, sc,wc);

monitorcoord.setY(ui->mychar->coords\_converter.y2+ui->mychar->offsety-monitorcoord.y());

monitorcoord.setX(monitorcoord.x()+ui->mychar->offsetx);

}

void MainWindow::resizeEvent(QResizeEvent \*r\_event)

{

}

void MainWindow::on\_horizontalSlider1\_valueChanged(int value)

{

// ui->mychar->offsetx = (value-50)\*ui->lineEdit->text().toFloat();

}

void MainWindow::on\_horizontalSlider\_2\_valueChanged(int value)

{

// ui->mychar->offsety = (value-50)\*ui->lineEdit->text().toFloat();

}

void MainWindow::on\_rangeEdit\_textEdited(const QString &arg1)

{

ui->mychar->a = arg1.split(",")[0].toFloat();

ui->mychar->b = arg1.split(",")[1].toFloat();

}

Main.cpp

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

