**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**  
**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2

Компьютерная графика

**Построение графиков функций**

Выполнил: ст. группы ВТ-22  
Ковалёв И. Д.

Проверил: Осипов О. В.

**Белгород 2019**

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

**Задания к работе:**

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

Содержимое файла axis.h:

#ifndef AXIS\_H

#define AXIS\_H

#include <QtMath>

#include <QVector>

#include <QPainter>

class Axis{

protected:

QPainter \* painter;

QRect scr;

double firstCoord, lastCoord;

int h, width;

public:

Axis(QPainter \* painter, const QRect& scr, double firstCoord, double lastCoord):

painter(painter), scr(scr), firstCoord(firstCoord), lastCoord(lastCoord){};

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

void draw(double h) {};

};

class AbscissaAxis : public Axis{

public:

AbscissaAxis(QPainter \* painter, const QRect& scr, double firstPoint, double lastPoint);

void draw(double h);

};

class OrdinateAxis : public Axis{

public:

OrdinateAxis(QPainter \* painter, const QRect& scr, double firstPoint, double lastPoint);

void draw(double h);

};

#endif // AXIS\_H

Содержимое файла chart.h:

#ifndef CHART\_H

#define CHART\_H

#include "axis.h"

#include "worldtoscreenconverter.h"

#include <QPainter>

#include <QWidget>

class **Chart** : public QWidget{

Q\_OBJECT

private:

double (\*f)(double x);

double x1, x2;

int h1;

public:

**Chart**(double(func)(double x), double x1, double x2, int h1, QWidget \*parent = 0);

void **setFunc**(double(func)(double x));

void **setDomainOfDef**(double x1, double x2);

void ***paintEvent***(QPaintEvent\* e);

public slots:

void **setX1**(double x);

void **setX2**(double x);

void **setH** (int h);

};

#endif // CHART\_H

Содержимое файла mainWindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QWidget>

#include <QObject>

#include <QWheelEvent>

#include <qlayout.h>

#include "chart.h"

#include <QSpinBox>

#include <QLineEdit>

#include <QLabel>

#include <QFormLayout>

class MainWindow : public QWidget

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private:

void wheelEvent(QWheelEvent\* wheelEvent);

Chart \* chart;

QSpinBox \* b;

QDoubleSpinBox \* x1val;

QDoubleSpinBox \* x2val;

QSpinBox \* hval;

QLabel \* x1l, \* x2l, \* hl, bl;

int h;

double x1, x2;

public slots:

void bValueChanged(int value);

void x1Changed(double value);

void x2Changed(double value);

void hChanged(int value);

};

#endif // MAINWINDOW\_H

Содержимое файла axis.cpp:

#include "axis.h"

#include <QtDebug>

void Axis::**step**(double& k, double& h, int& m, int& n){

double hm = h / k;

qDebug() << hm;

double n1 = log10(hm),

n2 = log10(hm / 2),

n5 = log10(hm / 5);

QVector <int> pows(6);

pows.push\_front(floor(n1));

pows.push\_front(ceil(n1));

pows.push\_front(floor(n2));

pows.push\_front(ceil(n2));

pows.push\_front(floor(n5));

pows.push\_front(ceil(n5));

QVector <double> diffs(6);

for (int i = 0; i < 6; i++){

if (i < 2) diffs[i] = fabs(pow(10, pows[i]) - hm);

else if (i > 3) diffs[i] = fabs(5 \* pow(10, pows[i]) - hm);

else diffs[i] = fabs(2 \* pow(10, pows[i]) - hm);

}

int minPos = 0;

double min = diffs[minPos];

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

if (min > diffs[i]){

min = diffs[i];

minPos = i;

}

}

n = pows[minPos];

if (minPos < 2) m = 1;

else if (minPos > 3) m = 5;

else m = 2;

double k1 = m \* pow (10, n);

h = k \* k1;

k = k1;

}

AbscissaAxis::**AbscissaAxis**(QPainter\* painter, const QRect& scr, double firstCoord, double lastCoord)

:Axis(painter, scr, firstCoord, lastCoord){}

void AbscissaAxis::**draw**(double h){

QPen oldPen = painter->pen(); // Сохраняем текущее перо для восстановления.

double k = (double)scr.width() / (lastCoord - firstCoord);

int m = 0;

int n = 0;

Axis::step(*k*, *h*, *m*, *n*);

qDebug() << h;

double firstWrd = ceil(firstCoord / k) \* k;

double firstScr = (firstWrd - firstCoord) \* h / k;

double currentScr= firstScr + scr.bottomLeft().x();

double currentWrd = firstWrd;

painter->setPen(QPen(Qt::black));

painter->drawLine(scr.bottomLeft(), scr.bottomRight());

for (; currentScr < scr.bottomRight().x() - 10; currentScr += h, currentWrd += k) {

painter->setPen(QPen(Qt::black));

QPointF p1(currentScr, scr.top());

QPointF p2(currentScr, scr.bottom() + 5);

painter->drawLine(p1, p2);

painter->setPen(QPen(Qt::black));

QString str;

str.setNum(currentWrd, 'g', 4);

painter->drawText(currentScr - h / 2 - 5, scr.bottom() + 10, h + 10, 20, Qt::AlignCenter, str);

}

QFont oldFont = painter->font();

QFont newFont;

newFont.setPixelSize(14);

painter->setFont(newFont);

painter->drawText(scr.right() - newFont.pixelSize() + 3, scr.bottom() + 2, newFont.pixelSize(), newFont.pixelSize(), Qt::AlignCenter, "X");

painter->setFont(oldFont);

painter->setPen(oldPen);

}

OrdinateAxis::**OrdinateAxis**(QPainter\* painter, const QRect& scr, double firstCoord, double lastCoord)

:Axis(painter, scr, firstCoord, lastCoord){}

void OrdinateAxis::**draw**(double h){

QPen oldPen = painter->pen();

double k = (double)scr.height() / (lastCoord - firstCoord);

int m = 0;

int n = 0;

Axis::step(*k*, *h*, *m*, *n*);

double firstWrd = ceil(firstCoord / k) \* k;

double firstScr = (firstWrd - firstCoord) \* h / k;

double currentScr = scr.bottomLeft().y() - firstScr;

double currentWrd = firstWrd;

painter->setPen(QPen(Qt::black));

painter->drawLine(scr.topLeft(), scr.bottomLeft());

for (; currentScr > scr.topLeft().y() + 20; currentScr -= h, currentWrd += k) {

painter->setPen(QPen(Qt::black));

QPointF p1(scr.left() - 5, currentScr);

QPointF p2(scr.right(), currentScr);

painter->drawLine(p1, p2);

QString str;

str.setNum(currentWrd, 'g', 4);

painter->drawText(scr.left() - 50, currentScr - h / 2, 40, h, Qt::AlignVCenter | Qt::AlignRight, str);

}

QFont oldFont = painter->font();

QFont font;

font.setPixelSize(14);

painter->setFont(font);

painter->drawText(scr.left() - font.pixelSize(), scr.top(), font.pixelSize(), font.pixelSize(), Qt::AlignCenter, "Y");

painter->setFont(oldFont);

painter->setPen(oldPen);

}

Содержимое файла chart.cpp:

#include "chart.h"

#include <QtDebug>

Chart::Chart(double(func)(double x), double x1, double x2, int h1, QWidget \*parent)

: QWidget(parent), f(func), x1(x1), x2(x2), h1(h1){}

void Chart::setFunc(double func(double)){

f = func;

}

void Chart::setDomainOfDef(double x1, double x2){

this->x1 = x1;

this->x2 = x2;

}

void Chart::setX1(double x){

this->x1 = x;

update();

}

void Chart::setX2(double x){

this->x2 = x;

update();

}

void Chart::setH(int h){

this->h1 = h;

update();

}

void Chart::paintEvent(QPaintEvent\* e){

if (x1 >= x2) return;

QPainter painter(this);

painter.setPen(Qt::blue);

double h = (x2 - x1) / this->width();

//qDebug() << h;

double y1 = f(x1);

double y2 = y1;

for (double x = x1 + h; x <= x2; x += h){

double y = f(x);

if (y1 > y) y1 = y;

if (y2 < y) y2 = y;

}

QRect scr(50, 0, this->width() - 70, this->height() - 50);

WorldToScreenConverter converter(scr, x1, x2, y1, y2);

QPoint p1 = converter.toScr(x1, f(x1));

QPoint p2;

for (double x = x1 + h; x < x2; x += h) {

p2 = converter.toScr(x, f(x));

painter.drawLine(p1, p2);

p1 = p2;

}

AbscissaAxis abAxis(&painter, scr, x1, x2);

abAxis.draw(h1);

OrdinateAxis orAxis(&painter, scr, y1, y2);

orAxis.draw(h1);

//qDebug() << h1;

}

Содержимое файла mainWindow.cpp:

#include "mainwindow.h"

double **f1**(double x){

return 2 \* x + 1;

}

double **f2**(double x){

return x \* x;

}

double **f3**(double x){

return 3 \* x \* -1;

}

MainWindow::**MainWindow**(QWidget \*parent) :QWidget(parent){

h = 70; x1 = -4.666; x2 = -4.3;

//QFormLayout \* l = new QFormLayout;

QVBoxLayout \* l = new QVBoxLayout;

QSpinBox \* b = new QSpinBox;

QDoubleSpinBox \* x1val = new QDoubleSpinBox;

QDoubleSpinBox \* x2val = new QDoubleSpinBox;

QSpinBox \* hval = new QSpinBox;

hval->setRange(0, 500);

hval->setValue(h);

x1val->setRange(-1000, 1000);

x2val->setRange(-1000, 1000);

x1val->setValue(x1);

x2val->setValue(x2);

b->setRange(1, 3);

chart = new Chart(f1, x1, x2, h);

l->addWidget(chart);

l->addWidget(b);

//QLabel \* bl = new QLabel;

//bl->setText("Функция:");

//l->addWidget(bl);

l->addWidget(x1val);

//QLabel \* x1l = new QLabel;

//x1l->setText("x1:");

l->addWidget(x1l);

l->addWidget(x2val);

l->addWidget(hval);

this->setLayout(l);

this->resize(1200, 600);

connect(b, SIGNAL(valueChanged(int)), this, SLOT(bValueChanged(int)));

connect(hval, SIGNAL(valueChanged(int)), this, SLOT(hChanged(int)));

connect(x1val, SIGNAL(valueChanged(double)), this, SLOT(x1Changed(double)));

connect(x2val, SIGNAL(valueChanged(double)), this, SLOT(x2Changed(double)));

}

MainWindow::~***MainWindow***(){

delete chart;

}

void MainWindow::***wheelEvent***(QWheelEvent \*wheelEvent){

if (wheelEvent->delta() > 0){

x1 = x1 + 0.5;

x2 = x2 - 0.5;

chart->setX1(x1);

chart->setX2(x2);

//chart->setH(int(x1 - x2 / 10));

} else {

x1 = x1 - 0.5;

x2 = x2 + 0.5;

chart->setX1(x1);

chart->setX2(x2);

//chart->setH(int(x1 - x2 / 10));

}

}

void MainWindow::**bValueChanged**(int value){

switch (value) {

case 1:

chart->setFunc(f1);

chart->update();

break;

case 2:

chart->setFunc(f2);

chart->update();

break;

case 3:

chart->setFunc(f3);

chart->update();

break;

}

}

void MainWindow::**x1Changed**(double value){

chart->setX1(value);

}

void MainWindow::**x2Changed**(double value){

chart->setX2(value);

}

void MainWindow::**hChanged**(int value){

chart->setH(value);

}

Содержимое файла worldtoscreenconverter.cpp:

#include "worldtoscreenconverter.h"

WorldToScreenConverter::WorldToScreenConverter(const QRect& screen, double x1, double x2, double y1, double y2){

\_screen = screen;

\_x1 = x1;

\_x2 = x2;

\_y1 = y1;

\_y2 = y2;

}

QPoint WorldToScreenConverter::toScr(double x, double y){

return QPoint(X(x), Y(y));

}

QPoint WorldToScreenConverter::toScr(const QPointF &w){

return this->toScr(w.x(), w.y());

}

QPointF WorldToScreenConverter::toWrd(int X, int Y){

return QPointF(x(X), y(Y));

}

QPointF WorldToScreenConverter::toWrd(const QPoint &s){

return this->toWrd(s.x(), s.y());

}

double WorldToScreenConverter::x(int X){

return ((double)(X - \_screen.left()) / \_screen.width() \* (\_x2 - \_x1)) + \_x1;

}

double WorldToScreenConverter::y(int Y){

return ((double)(\_screen.bottom() - Y) / \_screen.height() \* (\_y2 - \_y1)) + \_y1;

}

int WorldToScreenConverter::X(double x){

return \_screen.left() + ((x - \_x1) / (\_x2 - \_x1) \* \_screen.width());

}

int WorldToScreenConverter::Y(double y){

return \_screen.bottom() - ((y - \_y1) / (\_y2 - \_y1) \* \_screen.height()) + 1;

}

Содержимое файла worldtoscreenconverter.h

#ifndef WORLDTOSCREENCONVERTER\_H

#define WORLDTOSCREENCONVERTER\_H

#include <QMainWindow>

class WorldToScreenConverter{

QRect \_screen; // Экранная система координат

double \_x1, \_x2, \_y1, \_y2; // Мировая система координат

public:

WorldToScreenConverter(const QRect& screen, double x1, double x2, double y1, double y2);

QRectF getScreen() { return \_screen; }

//Преобразует координаты точки из мировой в экранную

QPoint toScr(double x, double y);

QPoint toScr(const QPointF& w);

//Преобразует координаты точки из экранной в мировую

QPointF toWrd(int X, int Y);

QPointF toWrd(const QPoint& s);

//Преобразует значение абсциссы Х из экранной системы координат в мировую

double x(int X);

//Преобразует значение ординаты Y из экранной системы координат в мировую

double y(int Y);

//Преобразует значение Х из мировой системы координат в мировую

int X(double x);

//Преобразует значение Y из мировой системы координат в мировую

int Y(double y);

};

#endif // WORLDTOSCREENCONVERTER\_H



