Алгоритмы и структуры данных

Отчёт по лабораторной работе No2

Подготовил мельников никита, епи-2-16

2018

Оглавление

[Задание на выполнение лабораторной работы №2 2](#_Toc510311923)

[Усложненный вариант задания на выполнение лабораторной работы №2 2](#_Toc510311924)

[Методические указания к выполнению лабораторной работы №2 2](#_Toc510311925)

[Требуемые особенности разрабатываемой компьютерной программы (простой вариант) 2](#_Toc510311926)

[Требуемые особенности разрабатываемой компьютерной программы (усложненный вариант) 2](#_Toc510311927)

[Краткая теория по теме лабораторной работы 3](#_Toc510311928)

[Метод половинного деления как метод поиска корней функции 3](#_Toc510311929)

[Метод решения 4](#_Toc510311930)

[Реализация алгоритма 4](#_Toc510311931)

[Структура программы 4](#_Toc510311932)

[Руководство программиста 5](#_Toc510311933)

[Файл chart.h 5](#_Toc510311934)

[Файл mainwindow.h 5](#_Toc510311935)

[Файл formulaclean.h 6](#_Toc510311936)

[Руководство пользователя со скриншотами 7](#_Toc510311937)

[Элементы управления 7](#_Toc510311938)

[Работа с графиком и полем вывода промежуточных значений 7](#_Toc510311939)

[Анализ результатов работы программы 8](#_Toc510311940)

[Листинг программы 9](#_Toc510311941)

[Файл main.cpp 9](#_Toc510311942)

[Файл mainwindow.h 9](#_Toc510311943)

[Файл mainwindow.cpp 10](#_Toc510311944)

[Функция нахождения корня уравнения 13](#_Toc510311945)

[Файл chart.h 14](#_Toc510311946)

[Файл chart.cpp 15](#_Toc510311947)

[Файл formulaclean.h 16](#_Toc510311948)

[Файл formulaclean.cpp 17](#_Toc510311949)

# Задание на выполнение лабораторной работы №2

Написать компьютерную программу, позволяющую:

1. строить график заданной математической функции (конкретные функции назначаются по индивидуальным вариантам);
2. находить корень заданной функции в пределах заданного пользователем диапазона методом дихотомии с заданной пользователем точностью и отображать найденный корень на графике функции;
3. определять, сколько раз было вычислено значение функции в процессе нахождения корня

## Усложненный вариант задания на выполнение лабораторной работы №2

Выполнение данного варианта (вместо выше приведенного) позволит набрать большее количество баллов.

Написать компьютерную программу, решающую следующую задачу и визуализирующую решение на экране:

Даны несколько кругов в правой координатной полуплоскости. Требуется найти положение прямой, проведенной из начала координат так, чтобы суммарные площади частей кругов выше и ниже прямой отличались не более чем на некоторую величину EPS. Если несколько кругов пересекаются, то площадь областей пересечения учитывается многократно (суммарная площадь двух кругов в этой задаче не зависит от того, пересекаются они или нет).

Результатом работы алгоритма поиска должно быть уравнение прямой.

## Методические указания к выполнению лабораторной работы №2

## Требуемые особенности разрабатываемой компьютерной программы (простой вариант)

На экране должен быть изображен график функции, границы интервала поиска и найденное значение корня функции.

## Требуемые особенности разрабатываемой компьютерной программы (усложненный вариант)

Величина EPS (требуемая точность) и количество кругов вводится пользователем. Для каждого запуска алгоритма решения предварительно выбираются случайным образом положение и размеры кругов.

Для количества кругов меньше 10 круги и прямая должны быть отображены на экране.

# Краткая теория по теме лабораторной работы

**Метод половинного деления** известен также как **метод бисекции**. В данном методе интервал делится ровно пополам.

Такой подход обеспечивает гарантированную сходимость метода независимо от сложности функции - и это весьма важное свойство. Недостатком метода является то же самое - метод никогда не сойдется быстрее, т.е. сходимость метода всегда равна сходимости в наихудшем случае.

Метод половинного деления:

1. Один из простых способов поиска *корней функции одного аргумента*.
2. Применяется для нахождения *значений действительно-значной функции*, определяемому по какому-либо критерию (это может быть сравнение на *минимум*, *максимум* или конкретное число).

## Метод половинного деления как метод поиска корней функции

Перед применением метода для поиска корней функции необходимо отделить корни одним из известных способов, например, графическим методом. Отделение корней необходимо в случае, если неизвестно на каком отрезке нужно искать корень.

Будем считать, что корень t функции f(x) = 0 отделён на отрезке [a, b]. Задача заключается в том, чтобы найти и уточнить этот корень методом половинного деления. Другими словами, требуется найти приближённое значение корня с заданной точностью \eps.

Пусть функция f непрерывна на отрезке [a, b], f(a)\*f(b) < 0, ε = 0,01, t ∈ [a, b] - единственный корень уравнения f(x) = 0, a ≤ t ≤ b.

(Мы не рассматриваем случай, когда корней на отрезке [a, b] несколько, то есть более одного. В качестве ε можно взять и другое достаточно малое положительное число, например, 0,001.)

Поделим отрезок [a, b] пополам. Получим точку , a < c < b и два отрезка [a, c], [c, b].

* Если f(x) = 0, то корень tt найден (t = c).
* Если нет, то из двух полученных отрезков [a, c] и [c, b] надо выбрать один [a1;b1] такой, что f(a1)\*f(b1) < 0, то есть
  + [a1; b1] = [a, c], если f(a)\*f(c) < 0 или
  + [a1; b1] = [c, b], если f(c)\*f(b) < 0.

Новый отрезок [a1; b1] делим пополам. Получаем середину этого отрезка  и так далее.

Для того, чтобы найти приближённое значение корня с точностью до ε > 0, необходимо остановить процесс половинного деления на таком шаге n, на котором  и вычислить . Тогда можно взять t ≈ x.

# Метод решения

## Реализация алгоритма

Для реализации алгоритма необходимы 3 переменные с плавающей запятой: для значений левой и правой границ отрезка, на котором находится корень уравнения и для значения середины этого отрезка. Далее организуется цикл с предусловием. Условие выполнения цикла: значение промежутка между левой и правой границами больше чем заданная точность. Тело цикла: переменной, хранящей середину отрезка, присваивается сумма значений левой и правой границ отрезка делёная на 2, затем, если произведение значений функции уравнения от середины и правой границы отрезка меньше нуля, то значение середины присваивается переменной со значением левой границы, иначе значение середины присваивается переменной со значением правой границы отрезка.

## Структура программы

Программа написана как графическое приложение, основанное на фреймворке Qt 5. В окне приложения имеются: область вывода графика функции, текстовое поле для вывода промежуточных значений во время выполнения функции поиска корня, спинбоксы для установки левой и правой границ для графика, спинбоксы для установки левой и правой границ отрезка, на котором находится корень уравнения, спинбокс для установки точности вычисления; также присутствует поле для ввода функции и 2 кнопки: для вывода графика и для запуска алгоритма поиска корня уравнения.

Вычисление функции производится с помощью класса QScriptEngine, предоставляемого фреймворком; функция выполняется классом как Qt Script[[1]](#footnote-1) код. Для возможности использования тригонометрических функций используется класс Math, предоставляемый Qt Script.

Более подробно о Qt Script можно узнать на официальном сайте фреймворка: <http://doc.qt.io/qt-5/qtscript-index.html>

# Руководство программиста

Здесь будут описаны только пользовательские классы, не являющиеся частью фреймворка Qt 5. Описание остальных классов, начинающихся с префикса “Q”(например QWidget), можно посмотреть на официальном сайте фреймворка: <http://doc.qt.io/>

## Файл chart.h

*class* Chart : *public* QChart

{

Q\_OBJECT

QString func; // хранит функцию в строковом виде

double LeftBorder; // хранит левую границу графика

double RightBorder;// хранит правую границу графика

double Step; // хранит шаг увеличения X на графике

*public*:

Chart(double leftBorder, double rightBorder, double step, *const* QString &f); // конструктор, принимающий значения левой и правой границ и стрковое представление функции

double **leftBorder**() *const*; // геттер для поля левой границы графика

double **rightBorder**() *const*;// геттер для поля правой границы графика

*public* *slots*:

void **setLeftBorder**(double value); // слот-сеттер для установки значения левой границы графика

void **setRightBorder**(double value); // слот-сеттер для установки значения правой границы графика

void **setF**(*const* QString &s); // слот-сеттер для установки функции

void **repaint**(); // слот для перерисовки графика

};

## Файл mainwindow.h

*class* MainWindow : *public* QWidget

{

Q\_OBJECT

Chart \*chart; // указатель на класс, хранящий график

QChartView \*chartView; // указатель на widget, представляющий график

QTextBrowser \*outputField; // указатель на текстовое поле, отображающее промежуточные значения функции нахождения корня уравнения

QToolButton \*startBut, // указатель на кнопку начала демонстрации работы алгоритма

\*updateBut; // указатель на кнопку обновления функции на графике

QLineEdit \*function; // текстовое поле, служащее для ввода функции

QString lastFunc; // хранит последнее, синтаксически верное, строковое представление функции

QDoubleSpinBox \*leftBorder, // спинбокс для установки левой границы графика

\*rightBorder, // спинбокс для установки правой границы графика

\*leftUserBorder, // спинбокс для установки левой границы отрезка, на котором находится корень уравнения

\*rightUserBorder, // спинбокс для установки правой границы отрезка, на котором находится корень уравнения

\*epsilon;

*private* *slots*:

void **onFunctionEditingFinished**(); // слот, отвечающий на нажатие кнопки обновления графика

*public*:

MainWindow(QWidget \*parent = 0);

~***MainWindow***();

*public* *slots*:

void **runAlgo**(); // функция нахождения корня уравнения

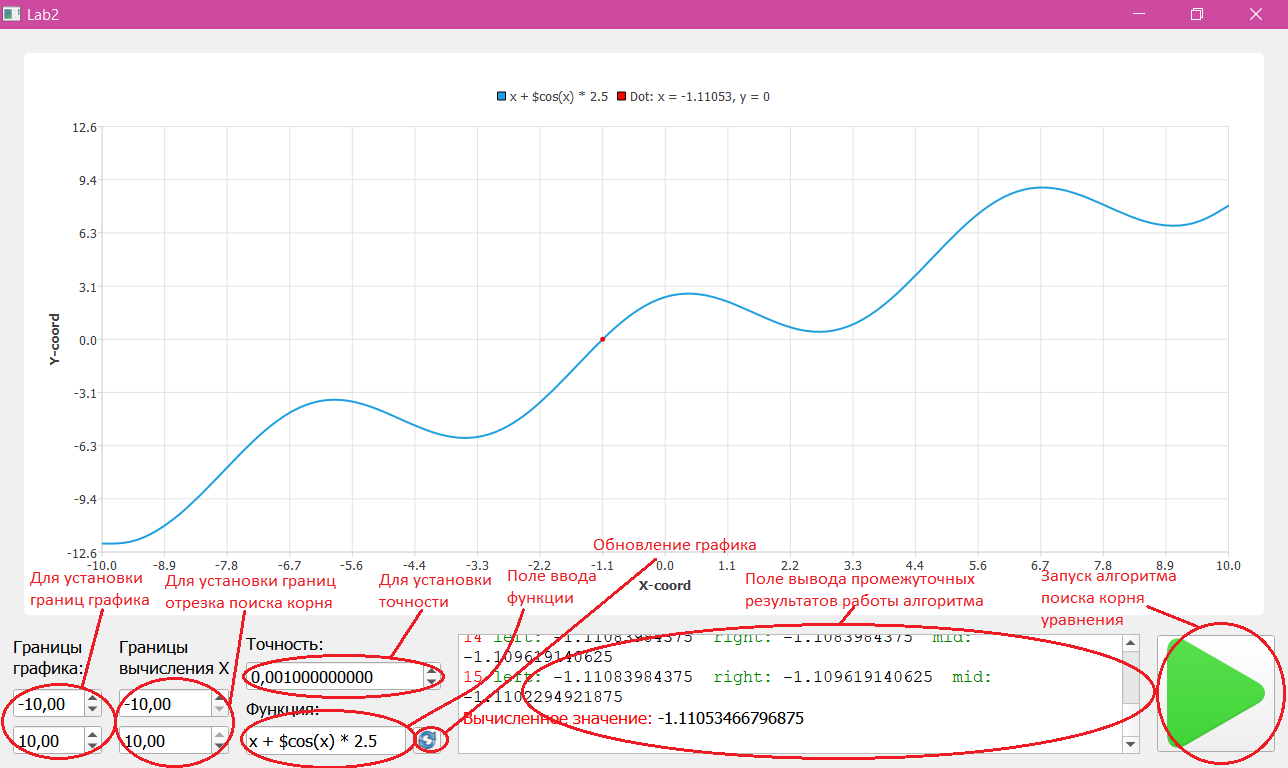
};

## Файл formulaclean.h

QString **formulaClean**(*const* QString &f); // функция подготовки строкового представления функции, вводимой пользователем, для последующего выполнения классом QScriptEngine

# Руководство пользователя со скриншотами

## Элементы управления



## Работа с графиком и полем вывода промежуточных значений



# Анализ результатов работы программы

|  |  |  |
| --- | --- | --- |
| Длинна отрезка, на котором находится корень | Точность | Количество итераций цикла |
| 20 | 0,001 | 15 |
| 16 | 0,001 | 14 |
| 12 | 0,001 | 14 |
| 8 | 0,001 | 13 |
| 4 | 0,001 | 12 |
| 20 | 0,000001 | 25 |
| 16 | 0,000001 | 24 |
| 12 | 0,000001 | 24 |
| 8 | 0,000001 | 23 |
| 4 | 0,000001 | 22 |
| 20 | 0,000000001 | 35 |
| 16 | 0,000000001 | 34 |
| 12 | 0,000000001 | 34 |
| 8 | 0,000000001 | 33 |
| 4 | 0,000000001 | 32 |
| 20 | 0,000000000001 | 45 |
| 16 | 0,000000000001 | 44 |
| 12 | 0,000000000001 | 44 |
| 8 | 0,000000000001 | 43 |
| 4 | 0,000000000001 | 42 |

Реализованный алгоритм поиска корня функции методом половинного деления имеет сложность , где a – длина отрезка, на котором находится искомый корень, а ε – точность. Это доказывает и сравнительный анализ, приведённый в таблице выше.

# Листинг программы

## Файл main.cpp

#include "mainwindow.h"

#include <QApplication>

#include <QStyleFactory>

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

{

QApplication a(argc, argv);

QApplication::setStyle(QStyleFactory::create("Fusion"));

MainWindow w;

w.setFont(QFont(QApplication::font().defaultFamily(), 10));

w.setGeometry(50, 100, 1000, 600);

w.showMaximized();

*return* a.exec();

}

## Файл mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QWidget>

#include <QChartView>

#include <QTextBrowser>

#include <QLabel>

#include <QDoubleSpinBox>

#include <QToolButton>

#include <QLineEdit>

#include <QScrollBar>

#include <QVBoxLayout>

#include <QHBoxLayout>

#include <QBitmap>

#include <QDir>

#include <QScriptEngine>

#include <QMessageBox>

#include <QStyle>

#include "chart.h"

#include "formulaclean.h"

*using* *namespace* QtCharts;

*class* MainWindow : *public* QWidget

{

Q\_OBJECT

Chart \*chart;

QChartView \*chartView;

QTextBrowser \*outputField;

QToolButton \*startBut,

\*updateBut;

QLineEdit \*function;

QString lastFunc;

QDoubleSpinBox \*leftBorder,

\*rightBorder,

\*leftUserBorder,

\*rightUserBorder,

\*epsilon;

*private* *slots*:

void **onFunctionEditingFinished**();

*public*:

MainWindow(QWidget \*parent = 0);

~***MainWindow***();

*public* *slots*:

void **runAlgo**();

};

#endif *//* *MAINWINDOW\_H*

## Файл mainwindow.cpp

#include "mainwindow.h"

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

QWidget(parent)

{

leftBorder = *new* QDoubleSpinBox();

rightBorder = *new* QDoubleSpinBox;

leftUserBorder = *new* QDoubleSpinBox();

rightUserBorder = *new* QDoubleSpinBox();

epsilon= *new* QDoubleSpinBox();

leftBorder->setRange(-100., 10.);

rightBorder->setRange(-10., 100.);

leftBorder->setValue(-10.);

rightBorder->setValue(10.);

leftUserBorder->setRange(leftBorder->value(), rightBorder->value());

rightUserBorder->setRange(leftBorder->value(), rightBorder->value());

leftUserBorder->setValue(leftBorder->value());

rightUserBorder->setValue(rightBorder->value());

epsilon->setDecimals(12);

epsilon->setRange(1e-12, 1.);

epsilon->setValue(0.001);

epsilon->setMaximumWidth(200);

leftBorder->setToolTip("Левая граница графика функции");

rightBorder->setToolTip("Правая граница графика функции");

leftUserBorder->setToolTip("Левая граница вычислений X");

rightUserBorder->setToolTip("Правая граница вычислений X");

epsilon->setToolTip("Точность вычислений");

function = *new* QLineEdit("x + $cos(x) \* 2.5");

function->setMaximumWidth(160);

function->setToolTip("Функция от X, например $cos(x) / 7.35 или $pow(x, 2) \* $cos(x\*$E)\n"

"(Использование \"$\" для тригонометрических и встроенных функций обязательно)\n"

"Доступные элементы-функции:\n"

"$abs(x)\n"

"$acos(x)\n"

"$asin(x)\n"

"$atan(x)\n"

"$atan2(y, x)\n"

"$ceil(x)\n"

"$cos(x)\n"

"$exp(x)\n"

"$floor(x)\n"

"$log(x)\n"

"$pow(x, y)\n"

"$random()\n"

"$round(x)\n"

"$sin(x)\n"

"$sqrt(x)\n"

"$tan(x)\n"

"$E"

"$PI\n"

);

lastFunc = function->text();

chart = *new* Chart(leftBorder->value(), rightBorder->value(), 0.5, function->text());

chartView = *new* QChartView(chart);

chartView->setRenderHint(QPainter::*Antialiasing*);

chartView->setRubberBand(QChartView::*RectangleRubberBand*);

outputField = *new* QTextBrowser();

outputField->setFixedHeight(120);

outputField->setVerticalScrollBarPolicy(Qt::*ScrollBarAsNeeded*);

outputField->setFont(QFont("Monospace"));

startBut = *new* QToolButton();

startBut->setIcon(QIcon(QPixmap(":/run.ico")));

startBut->setIconSize(QSize(110, 110));

startBut->setToolTip("Запустить выполнение алгоритма");

updateBut = *new* QToolButton();

updateBut->setIcon(style()->*standardIcon*(QStyle::*SP\_BrowserReload*));

updateBut->setToolTip("Нарисовать график");

connect(leftBorder, *static\_cast*<void(QDoubleSpinBox::\*)(double)>(&QDoubleSpinBox::valueChanged), [=](double var){

rightBorder->setMinimum(var);

leftUserBorder->setMinimum(var);

chart->setLeftBorder(var);

});

connect(rightBorder,

*static\_cast*<void(QDoubleSpinBox::\*)(double)>(&QDoubleSpinBox::valueChanged), [=](double var){

leftBorder->setMaximum(var);

rightUserBorder->setMaximum(var);

chart->setRightBorder(var);

});

connect(leftUserBorder,

*static\_cast*<void(QDoubleSpinBox::\*)(double)>(&QDoubleSpinBox::valueChanged),

rightUserBorder,

&QDoubleSpinBox::setMinimum);

connect(rightUserBorder,

*static\_cast*<void(QDoubleSpinBox::\*)(double)>(&QDoubleSpinBox::valueChanged),

leftUserBorder,

&QDoubleSpinBox::setMaximum);

connect(startBut, &QToolButton::clicked, *this*, &MainWindow::runAlgo);

connect(updateBut, &QToolButton::clicked, *this*, &MainWindow::onFunctionEditingFinished);

QVBoxLayout \*vBorderLay = *new* QVBoxLayout();

vBorderLay->addWidget(*new* QLabel("Границы\nграфика:"));

vBorderLay->addWidget(leftBorder);

vBorderLay->addWidget(rightBorder);

QVBoxLayout \*vUserBorderLay = *new* QVBoxLayout();

vUserBorderLay->addWidget(*new* QLabel("Границы\nвычисления X"));

vUserBorderLay->addWidget(leftUserBorder);

vUserBorderLay->addWidget(rightUserBorder);

QHBoxLayout \*hFuncLay = *new* QHBoxLayout();

hFuncLay->addWidget(function);

hFuncLay->addWidget(updateBut);

QVBoxLayout \*vFuncEpsLay = *new* QVBoxLayout();

vFuncEpsLay->addWidget(*new* QLabel("Точность:"));

vFuncEpsLay->addWidget(epsilon);

vFuncEpsLay->addWidget(*new* QLabel("Функция: "));

vFuncEpsLay->addLayout(hFuncLay);

QHBoxLayout \*hBorderLay = *new* QHBoxLayout();

hBorderLay->addLayout(vBorderLay);

hBorderLay->addSpacing(10);

hBorderLay->addLayout(vUserBorderLay);

hBorderLay->addSpacing(10);

hBorderLay->addLayout(vFuncEpsLay);

hBorderLay->addSpacing(10);

hBorderLay->addWidget(outputField);

hBorderLay->addSpacing(10);

hBorderLay->addWidget(startBut);

QVBoxLayout \*vlay = *new* QVBoxLayout();

vlay->addWidget(chartView);

vlay->addLayout(hBorderLay);

setLayout(vlay);

}

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

{

}

void MainWindow::**onFunctionEditingFinished**()

{

*if* (chart->series().size() > 1)

{

QSplineSeries \*series = *static\_cast*<QSplineSeries\*>(chart->series()[1]);

*delete* series;

}

function->clearFocus();

QString func = function->text();

QScriptEngine eng;

bool ok;

eng.evaluate(formulaClean(func).arg(0)).toVariant().toDouble(&ok);

*if* (!ok)

{

QMessageBox::warning(*this*, "Ошибка!", QString("Невозможно посчитать формулу:\n%1").arg(function->text()));

function->setText(lastFunc);

*return*;

}

chart->setF(func);

lastFunc = func;

}

## Функция нахождения корня уравнения

void MainWindow::**runAlgo**()

{

double a1 = leftUserBorder->value(), // левая границ поиска

a2 = rightUserBorder->value(), // правая граница поиска

eps = epsilon->value(), // точность

mid; // середина

QString func = formulaClean(function->text()); // подготовка текстового представления функции к выполнения классом QScriptEngine

QScriptEngine eng;

bool ok;

eng.evaluate(func.arg(0)).toVariant().toDouble(&ok); // проверка текстового представления функции на валидность

*if* (!ok) // если проверка не пройдена

{

QMessageBox::warning(*this*, "Ошибка!", QString("Невозможно посчитать формулу:\n%1").arg(function->text()));

*return*;

}

// вывод заголовка для промежуточных значений

outputField->append(

QString("Фунцкия: <span style='color: red'>%1</span><br>"

"Tочность: <span style='color: red'>%2</span>")

.arg(function->text())

.arg(epsilon->value(), 0, 'e', 0));

// определение макроса, высчитывающего значение функции

#define f(x) eng.evaluate(func.arg(x)).toNumber()

int i = 1; // счётчик колличества итераций цикла

*while* (a2 - a1 > eps)

{

mid = (a1 + a2) / 2;

// вывод промежуточных значений в текстовое поле

outputField->append(QString(

"<pre>"

"<span style='color: red'>%1 </span>"

"<span style='color: green'>left: </span>%2 "

"<span style='color: green'>right: </span>%3 "

"<span style='color: green'>mid: </span>%4"

"</pre>"

).arg(i++).arg(a1, 0, 'g', 15).arg(a2, 0, 'g', 15).arg(mid, 0, 'g', 15));

*if*( f(a2) \* f(mid) < 0.)

a1 = mid;

*else*

a2 = mid;

}

#undef f

mid = (a1 + a2) / 2.;

// вывод вычисленного значения в текстовое поле

outputField->append(QString("<span style='color: red'>Вычисленное значение:</span> %1<br>").arg(mid, 0, 'g', 15));

outputField->verticalScrollBar()->setValue(outputField->verticalScrollBar()->maximum());

QLineSeries \*dot;

*if* (chart->series().size() > 1)

{

dot = *static\_cast*<QLineSeries\*>(chart->series().at(1));

dot->clear();

dot->setName(QString("Dot: x = %1, y = 0").arg(mid));

}

*else*

{

dot = *new* QLineSeries();

dot->*setPen*(QPen(QBrush(Qt::*red*), 5, Qt::*SolidLine*, Qt::*RoundCap*));

dot->setName(QString("Dot: x = %1, y = 0").arg(mid));

chart->addSeries(dot);

}

dot->append(mid-0.0001, 0.);

dot->append(mid+0.0001, 0.);

chart->repaint();

}

## Файл chart.h

#ifndef CHART\_H

#define CHART\_H

#include <QObject>

#include <QChart>

#include <QSplineSeries>

#include <QValueAxis>

#include <QScriptEngine>

#include "formulaclean.h"

*using* *namespace* QtCharts;

*class* Chart : *public* QChart

{

Q\_OBJECT

QString func;

double LeftBorder;

double RightBorder;

double Step;

*public*:

Chart(double leftBorder, double rightBorder, double step, *const* QString &f);

double **leftBorder**() *const*;

double **rightBorder**() *const*;

*public* *slots*:

void **setLeftBorder**(double value);

void **setRightBorder**(double value);

void **setF**(*const* QString &s);

void **repaint**();

};

#endif *//* *CHART\_H*

## Файл chart.cpp

#include "chart.h"

Chart::**Chart**(double leftBorder, double rightBorder, double step, *const* QString &f) :

func(f),

LeftBorder(leftBorder),

RightBorder(rightBorder),

Step(step)

{

repaint();

}

void Chart::**repaint**()

{

QSplineSeries \*ser;

*if* (series().isEmpty())

{

ser = *new* QSplineSeries();

}

*else*

{

ser = *static\_cast*<QSplineSeries\*>(series().at(0));

ser->clear();

}

ser->setName(func);

QScriptEngine eng;

QString fun = formulaClean(func);

double min = eng.evaluate(fun.arg(LeftBorder)).toNumber();

double max = min;

*for* (double x = LeftBorder; x <= RightBorder; x+=Step)

{

double y = eng.evaluate(fun.arg(x)).toNumber();

ser->append(x, y);

*if* (y < min) min = y;

*if* (y > max) max = y;

}

max = (qAbs(max) > qAbs(min)) ? qAbs(max) : qAbs(min);

*if* (series().isEmpty()) addSeries(ser);

createDefaultAxes();

*static\_cast*<QValueAxis\*>(axisX(ser))->setRange(LeftBorder, RightBorder);

*static\_cast*<QValueAxis\*>(axisY(ser))->setRange(-max-0.5, max+0.5);

*static\_cast*<QValueAxis\*>(axisX(ser))->setTitleText("X-coord");

*static\_cast*<QValueAxis\*>(axisY(ser))->setTitleText("Y-coord");

*static\_cast*<QValueAxis\*>(axisX(ser))->setTickCount(19);

*static\_cast*<QValueAxis\*>(axisY(ser))->setTickCount(9);

}

double Chart::**leftBorder**() *const*

{

*return* LeftBorder;

}

void Chart::**setLeftBorder**(double value)

{

LeftBorder = value;

repaint();

}

double Chart::**rightBorder**() *const*

{

*return* RightBorder;

}

void Chart::**setRightBorder**(double value)

{

RightBorder = value;

repaint();

}

void Chart::**setF**(*const* QString &s)

{

func = s;

repaint();

}

## Файл formulaclean.h

#ifndef FORMULACLEAN\_H

#define FORMULACLEAN\_H

#include <QString>

QString **formulaClean**(*const* QString &f);

#endif *//* *FORMULACLEAN\_H*

## Файл formulaclean.cpp

#include "formulaclean.h"

QString **formulaClean**(*const* QString &f)

{

QString func = f;

*return* func.replace("$", "Math.").replace("exp", "#").replace("x", "%1", Qt::*CaseInsensitive*).replace("#", "exp");

}

1. Скриптовый язык, основанный на ECMA-262 стандарте [↑](#footnote-ref-1)