**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа №1**

по курсу «Компьютерная графика»

Тема: «Построение изображений 2D-кривых»

Студент: Мариничев И. А.

Группа: М8О-308Б-19

Преподаватель: Филиппов Г. С.

Оценка:

Москва

2021

**1. Постановка задачи.**

Написать и отладить программу, строящую изображение заданной замечательной кривой.

**Вариант №8**: , где x, y — декартовы координаты, a (a>0), B — константы, значение которых выбирается пользователем (вводится в окне программы).

Обеспечить автоматическое масштабирование и центрирование кривой при изменении размеров окна.

**2. Описание программы.**

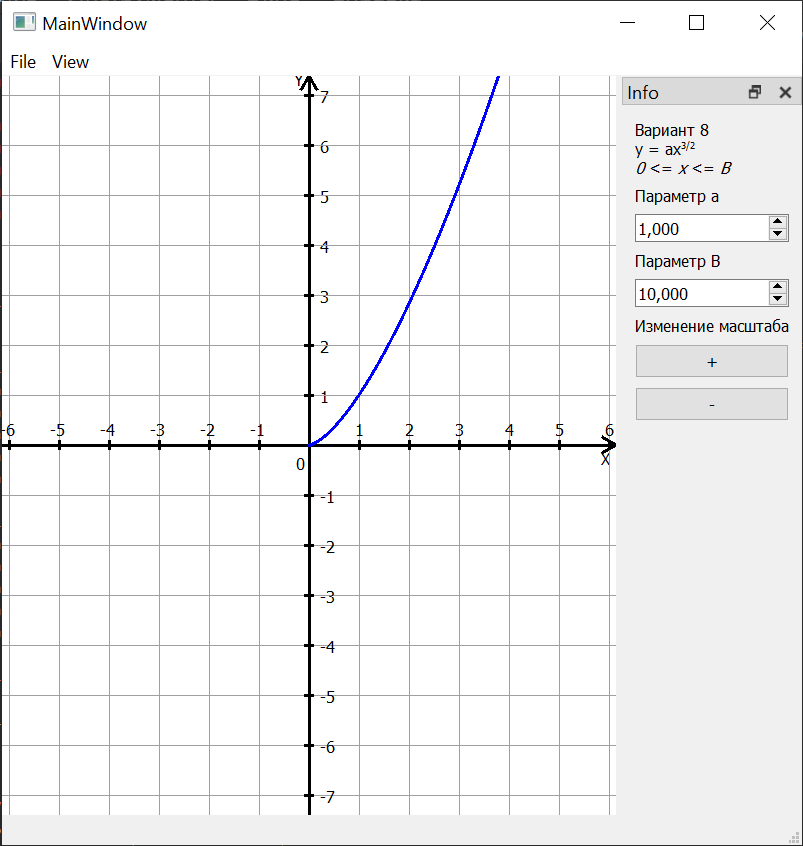
Для решения задачи я решил использовать C++ и фреймворк Qt, в котором использовал библиотеку QPainter для отрисовки точек и линий.

Создаётся координатная плоскость, на которой отрисовывается график, предлагаемый по умолчанию, со значением констант a = 1,000, B = 10,000 и добавляется панель для ввода значений параметра. Эта панель изначально находится справа, но ее можно перемещать влево, вверх, вниз или вообще вынести в отдельное окно.

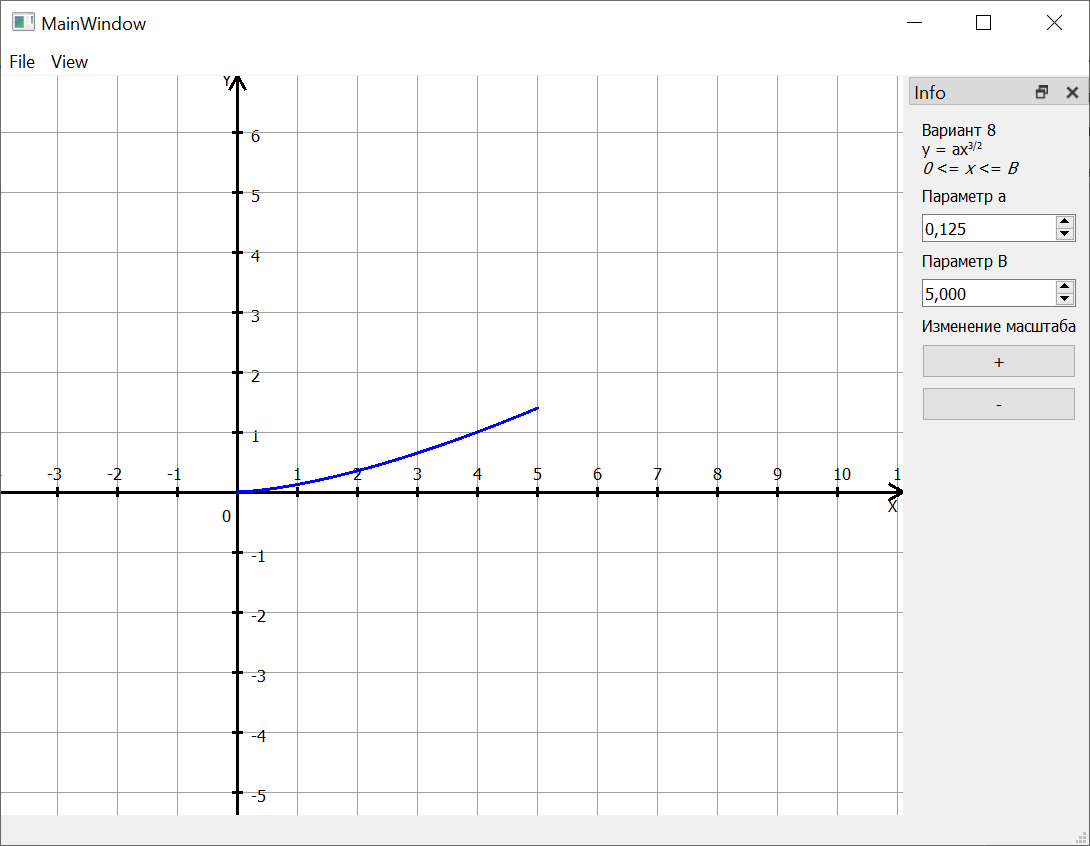
При изменении размера окна вызывается функция resizeEvent(), которая вычисляет коэффициенты изменения ширины и высоты окна, а затем при учете этих коэффициентов график перерисовывается. Таким образом реализуется автоматическое масштабирование. Также реализовано перемещение по координатной плоскости при зажатии ЛКМ, также возможно изменение масштаба при помощи колеса мыши или соответствующих кнопок на панели.

**3. Демонстрация работы программы.**

1) Значения параметров a = 1; B = 10



2) Значения параметров a = 0,125; B = 5. Применено масштабирование

****

**4. Основной код программы.**

1) Отрисовка графика функции

void **draw\_function**(widget \*w) {

QPainter ptr{w};

ptr.setPen(QPen(Qt::*blue*, 3));

*const* double param\_a = w->uPanel\_pointer->parameter\_a();

*const* double param\_B = w->uPanel\_pointer->parameter\_B();

*const* double step = 0.01;

*const* double x\_unit\_segment = 1.; *//* *value* *of* *x* *coordinate* *division*

*const* double y\_unit\_segment = 1.; *//* *value* *of* *y* *coordinate* *division*

QPointF p1{w->center\_x, w->center\_y};

QPointF p2{};

*for* (double x = 0.; x <= param\_B; x += step) {

p2 = {x \* (w->step\_x / x\_unit\_segment) + w->center\_x, w->center\_y - (param\_a \* pow(x, 3. / 2.)) \* (w->step\_y / y\_unit\_segment)};

ptr.drawLine(p1, p2);

p1 = p2;

}

}

2) Отрисовка системы координат, сетки и шкалы со значениями

void **draw\_coord\_system**(widget \*w) {

*const* int x\_unit\_segment = 1.; *//* *value* *of* *x* *coordinate* *division*

*const* int y\_unit\_segment = 1.; *//* *value* *of* *y* *coordinate* *division*

*const* double pi = atan(1) \* 4;

QPainter ptr{w};

ptr.setPen(QPen(Qt::*black*, 3));

*//* *drawing* *X* *axis*

QPoint p1{0, *static\_cast*<int>(w->center\_y)};

QPoint p2{w->width(), *static\_cast*<int>(w->center\_y)};

*const* int branch\_len = 15;

ptr.drawLine(p1, p2);

*//* *drawing* *'>'* *at* *the* *end* *of* *X* *axis*

QPointF p\_branch1{*static\_cast*<double>(w->width()), *static\_cast*<double>(w->center\_y)};

QPointF p\_branch2{branch\_len \* cos(-5 \* pi / 6) + w->width(),

branch\_len \* sin(-5 \* pi / 6) + w->center\_y};

ptr.drawLine(p\_branch1, p\_branch2);

p\_branch2 = {branch\_len \* cos(5 \* pi / 6) + w->width(),

branch\_len \* sin(5 \* pi / 6) + w->center\_y};

ptr.drawLine(p\_branch1, p\_branch2);

*//* *drawing* *Y* *axis*

p1.setX(*static\_cast*<int>(w->center\_x));

p1.setY(0);

p2.setX(*static\_cast*<int>(w->center\_x));

p2.setY(w->height());

ptr.drawLine(p1, p2);

*//* *drawing* *'>'* *at* *the* *end* *of* *y* *axis*

p\_branch1 = {*static\_cast*<double>(w->center\_x), 0};

p\_branch2 = {branch\_len \* cos(pi / 3) + w->center\_x,

branch\_len \* sin(pi / 3)};

ptr.drawLine(p\_branch1, p\_branch2);

p\_branch2 = {branch\_len \* cos(2 \* pi / 3) + w->center\_x,

branch\_len \* sin(2 \* pi / 3)};

ptr.drawLine(p\_branch1, p\_branch2);

*//* *drawing* *'0'* *and* *'X'* *with* *'Y'*

ptr.setPen(QPen(Qt::*black*, 3));

ptr.drawText(QPointF(w->center\_x - (w->step\_x / 4), w->center\_y + (w->step\_y / 2)),

QString::number(0));

ptr.drawText(w->width() - 15, *static\_cast*<int>(w->center\_y + 20), "X");

ptr.drawText(*static\_cast*<int>(w->center\_x - 15), 10, "Y");

*//* *drawing* *grid*

p\_branch1.setY(w->center\_y + w->step\_y / 16);

p\_branch2.setY(w->center\_y - w->step\_y / 16);

p1.setY(0);

p2.setY(w->height());

*for* (int x = *static\_cast*<int>(w->step\_x), num = 0; x + w->center\_x < w->width() || w->center\_x - x > 0;

x += w->step\_x, num += x\_unit\_segment) {

*//* *drawing* *vertical* *lines* *of* *the* *grid* *(right* *half)*

ptr.setPen(Qt::*gray*);

p1.setX(*static\_cast*<int>(x + w->center\_x));

p2.setX(*static\_cast*<int>(x + w->center\_x));

ptr.drawLine(p1, p2);

*//* *drawing* *scale* *on* *positive* *X* *axis*

ptr.setPen(QPen(Qt::*black*, 3));

p\_branch1.setX(x + w->center\_x);

p\_branch2.setX(x + w->center\_x);

ptr.drawLine(p\_branch1, p\_branch2);

*//* *drawing* *scale* *numbers* *on* *positive* *X* *axis*

ptr.drawText(QPointF(x + w->center\_x - w->step\_x / 16, *static\_cast*<int>(p\_branch1.y()) - w->step\_y / 4),

QString::number(num + x\_unit\_segment));

*//* *drawing* *vertical* *lines* *of* *the* *grid* *(left* *half)*

ptr.setPen(Qt::*gray*);

p1.setX(*static\_cast*<int>(w->center\_x - x));

p2.setX(*static\_cast*<int>(w->center\_x - x));

ptr.drawLine(p1, p2);

*//* *drawing* *scale* *on* *negative* *X* *axis*

ptr.setPen(QPen(Qt::*black*, 3));

p\_branch1.setX(w->center\_x - x);

p\_branch2.setX(w->center\_x - x);

ptr.drawLine(p\_branch1, p\_branch2);

*//* *drawing* *scale* *numbers* *on* *negative* *X* *axis*

ptr.drawText(QPointF(w->center\_x - x - w->step\_x / 6, *static\_cast*<int>(p\_branch1.y()) - w->step\_y / 4),

QString::number(-1 \* (num + x\_unit\_segment)));

}

p\_branch1.setX(w->center\_x + w->step\_x / 16);

p\_branch2.setX(w->center\_x - w->step\_x / 16);

p1.setX(0);

p2.setX(w->width());

*for* (int y = *static\_cast*<int>(w->step\_y), num = 0; y + w->center\_y < w->height() || w->center\_y - y > 0;

y += w->step\_y, num +=y\_unit\_segment) {

*//* *drawing* *horizontal* *lines* *of* *the* *grid* *(lower* *half)*

ptr.setPen(Qt::*gray*);

p1.setY(*static\_cast*<int>(y + w->center\_y));

p2.setY(*static\_cast*<int>(y + w->center\_y));

ptr.drawLine(p1, p2);

*//* *drawing* *scale* *on* *negatie* *Y* *axis*

ptr.setPen(QPen(Qt::*black*, 3));

p\_branch1.setY(y + w->center\_y);

p\_branch2.setY(y + w->center\_y);

ptr.drawLine(p\_branch1, p\_branch2);

*//* *drawing* *scale* *numbers* *on* *negative* *Y* *axis*

ptr.drawText(QPointF(*static\_cast*<int>(p\_branch1.x()) + w->step\_x / 6, w->center\_y - y + w->step\_y / 6),

QString::number(num + y\_unit\_segment));

*//* *drawing* *horizontal* *lines* *of* *the* *grid* *(upper* *half)*

ptr.setPen(Qt::*gray*);

p1.setY(*static\_cast*<int>(w->center\_y - y));

p2.setY(*static\_cast*<int>(w->center\_y - y));

ptr.drawLine(p1, p2);

*//* *drawing* *scale* *on* *positive* *Y* *axis*

ptr.setPen(QPen(Qt::*black*, 3));

p\_branch1.setY(w->center\_y - y);

p\_branch2.setY(w->center\_y - y);

ptr.drawLine(p\_branch1, p\_branch2);

*//* *drawing* *scale* *numbers* *on* *positive* *Y* *axis*

ptr.drawText(QPointF(*static\_cast*<int>(p\_branch1.x()) + w->step\_x / 6, w->center\_y + y + w->step\_y / 6),

QString::number(-1 \* (num + y\_unit\_segment)));

}

}

3) Автоматическое масштабирование и центрирование кривой при изменении размеров окна

void widget::***resizeEvent***(QResizeEvent \*event) {

*if* (event->oldSize().width() == -1 || event->oldSize().height() == -1)

*return*;

double coef\_x = width() / *static\_cast*<double>(event->oldSize().width());

double coef\_y = height() / *static\_cast*<double>(event->oldSize().height());

center\_x \*= coef\_x;

center\_y \*= coef\_y;

update();

}

4) Вызов функций отрисовки графика и координат, а также задание координат центра при первом вызове

void widget::**paintEvent**(QPaintEvent \*) {

if (uPanel\_pointer == nullptr) {

return;

}

if (is\_first) {

center\_x = width() / 2;

center\_y = height() / 2;

is\_first = false;

}

draw\_coord\_system(this);

draw\_function(this);

}

**5. Выводы.**

В ходе данной лабораторной работы я познакомился с фреймворком Qt, при помощи которого смог отрисовать график 2-D кривой. Надеюсь, приобретенные навыки помогут мне в дальнейшем, уже при работе с трехмерными фигурами.