Лабораторная работа №2

Построение графиков функций

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

Теоретические сведения

1. **Экранная и мировая системы координат**

Для построения графиков функций необходимо оперировать двумя системами координат: мировой системой координат и экранной системой координат.

Экранная система координат – система координат графического компонента (окна приложения). Координатами точки в экранной системе координат является номер пикселя вдоль оси и номер строки пикселей вдоль оси . Координаты , могут меняться в следующих пределах:

Размеры экранной системы координат зависят от разрешения экрана и размеров окна. К примеру, при разрешении Full HD (1920×1080) , . Начало экранной системы координат расположено в левом верхнем углу экрана (окна).

*Y*2

*Y*1

(0, 0)

(*x*2, *y*2)

(*x*1, *y*1)

(*x*2, *y*1)

(*x*1, *y*2)

*X*

*Y*

*X*1

*X*2

*x*

*y*

*y*=*f*(*x*)

Рис. 1. Экранная и мировая системы координат

Параметры экранной системы координат (максимальное число пикселей в строке *X*max и максимальное число строк пикселей *Y*max) зависят от размера окна.

Вторая система координат – так называемая мировая или математическая. Она представляет собой декартову систему (*x*, *y*), определяемую программистом, и является независимой от конкретного графического устройства:

*x*1 < *x* < *x*2, *y*1 *< y < y*2.

Параметры, которыми задаются диапазоны изменения *x* и *y* (*x*1, *y*1, *x*2, *y*2), определяют прямоугольную область в математическом двумерном пространстве. Эти параметры зависят только от конкретной задачи, а единицей измерения является реальная физическая величина (метр, секунда или др.).

Мировые координаты и координаты устройства связаны между собой простыми соотношениями:

1. **Вычисление шагов разметки координатных осей**

Для задания шага разметки введём понятие – ***предпочтительный шаг*** (*H*) разметки в экранной системе координат. Это расстояние между двумя соседними разметками (рисками) координатной оси, измеряемое в пикселях. После расчёта действительного шага разметки он в общем случае не будет равен предпочтительному шагу (но будет близок к нему), потому что в мировой системе координат он будет выбираться из 1×10*n*, 2×10*n*, 5×10*n*. Смысл заключается в том, что нужно выбрать шаг таким образом, чтобы при пересчёте его в экранные координаты он был наиболее близок к *H*.

*Пример* 1. Пусть пикселей, область определения , ширина поля вывода *W* = 1500 пикселей. Вычислим действительный шаг.

На 100 пикселей приходит ся единиц в мировой системе координат. Эта величина наиболее близка к *h*=1×101. Пересчитаем *h* в экранной системе координат и получим действительный шаг:

Важно, чтобы координаты разметок были кратны *h*. В нашем примере диапазон значений *x* начинается с 15, *h*=10. Размечать оси значениями 15, 25, 35,… было бы неправильным. Оси должны быть размечены (рис. 2), начиная со значения

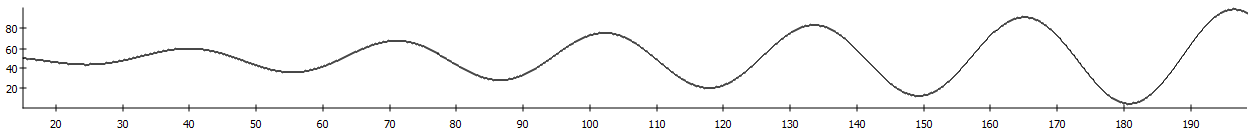


Рис. 2. Разметка оси при

*Пример* 2. Пусть .

На 100 пикселей приходится единиц в мировой системе координат. Оптимальным шагом разметки в данном случае будет единиц (рис. 3). Координаты первой разметки:

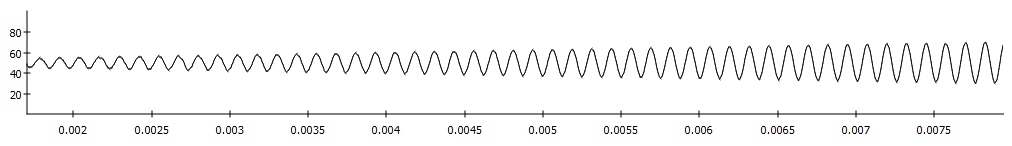


Рис. 3. Разметка оси при

Самый простой линейный алгоритм расчёта шага разметки заключается в следующем:

1. Пересчитать размер шага *H* в мировой системе координат:
2. Решить уравнения с неизвестными :
3. Выбрать шаг разметки в мировой системе координат, вычислив минимум из следующих 6 значений:
4. Вычислить шаг разметки в экранной системе координат:

Для расчёта шагов разметки необходима отдельная функция со следующей спецификацией:

1. Заголовок: void step(double& k, double& H, int& m, int& n).
2. Назначение: вычисляет шаг разметки в экранной и мировой системах координат.
3. Входные параметры: – количество пикселей на одну единицу в мировой системе координат; – предпочтительный шаг разметки.
4. Выходные параметры: – действительный шаг разметки в экранной системе координат, близкий к предпочтительному; ; – показатель степени; – действительный шаг разметки в мировой системе координат.
5. **Cтруктура классов приложения**
   1. Класс для преобразования мировых координат в экранные и наоборот

class WorldToScreenConverter

{

public:

// Экранная система координат

QRect screen;

// 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);

...

}

* 1. Классы Axis (ось), AbscissaAxis (ось абсцисс), OrdinateAxis (ось ординат) для рисования осей и их разметки.

class Axis

{

public:

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

...

};

class AbscissaAxis : Axis

{

public:

void draw(…);

...

};

class OrdinateAxis : Axis

{

public:

void draw(…);

...

};

* 1. Класс CurveData для хранения информации о выводимой на график функции , цвете, толщине кривой и других необходимых параметров.

class CurveData

{

QColor curveColor; // Цвет кривой

QPen curvePen; // Стиль пера

public:

...

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

};

От класса CurveData наследуются классы для хранения конкретных функций. Например, создадим класс для хранения синусоиды:

class SinCurveData : CurveData

{

public:

double *F*(double x) override

{

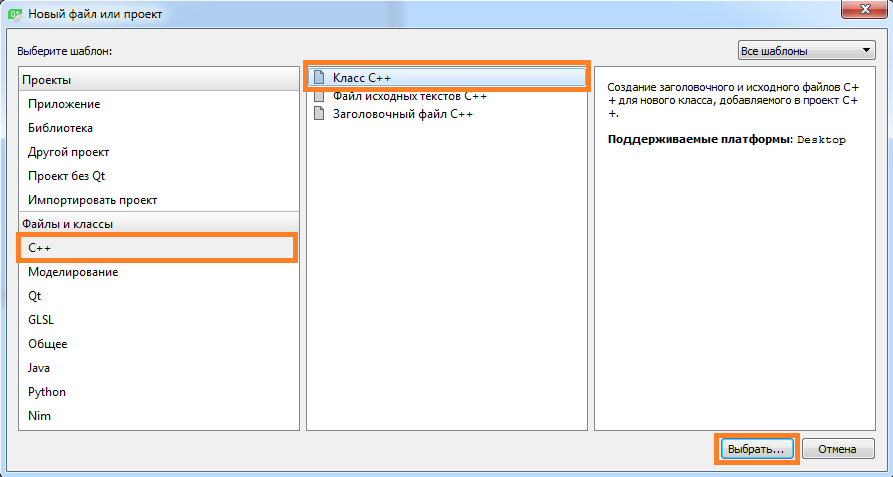
return sin(x);

}

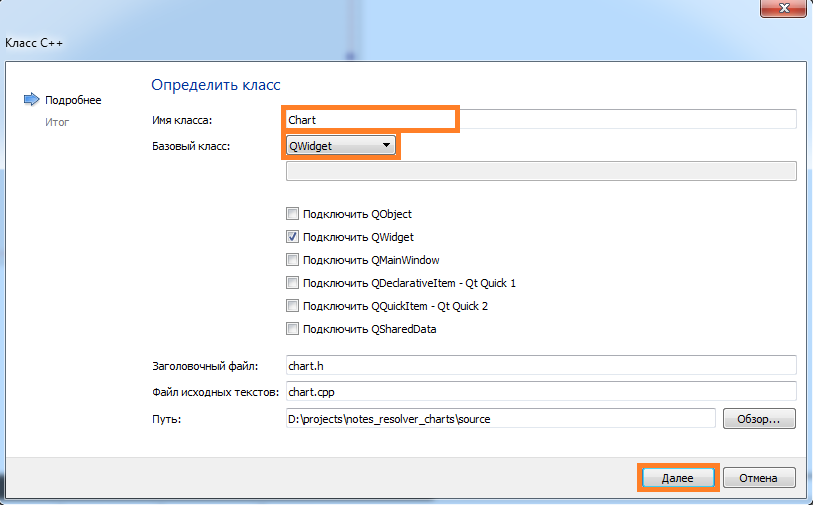
}

* 1. Основной класс Chart (график).

Объекты данного класса будут размещаться на оконной форме и являться графическими компонентами, такими же как кнопки, меню и др. Для создания класса-наследника от QWidget в Qt необходимо вызвать соответствующее диалоговое окно (сочетание клавиш **Ctrl+N**):



Указать название класса и базовый класс:



class Chart : public QWidget

{

Q\_OBJECT

public:

explicit Chart(QWidget \*parent = 0);

// 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);

private:

// Объект для преобразования мировых координат в экранные и наоборот

WorldToScreenConverter coords\_converter;

void *paintEvent*(QPaintEvent\* p\_event);

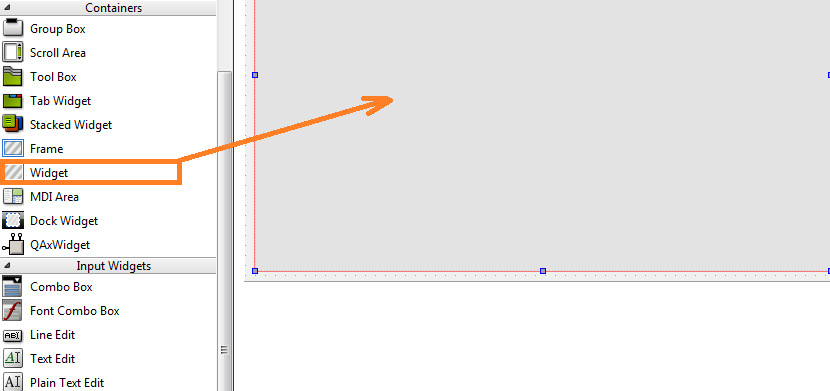
void *resizeEvent*(QResizeEvent\* r\_event);

...

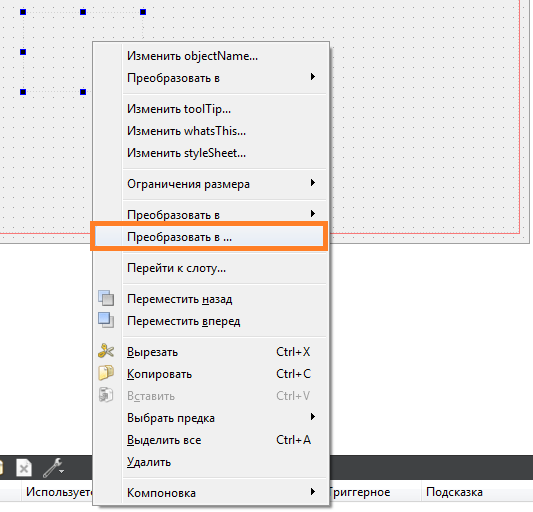
};

Так как основной класс наследуется от QWidget, то для размещения его на экранной форме необходимо выполнить следующие действия:

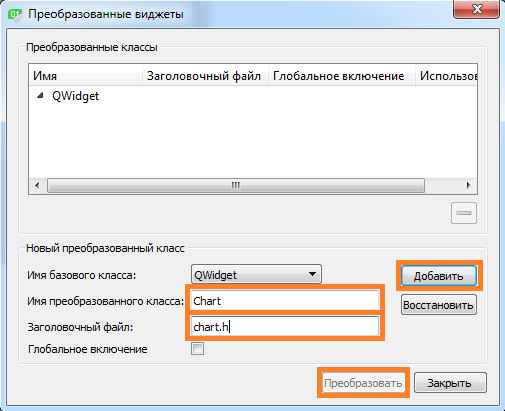
1. Поместить объект типа QWidget на форму:



1. Выделить данный объект и выбрать пункт «Преобразовать в …» контекстного меню:



1. Задать имя класса, к которому будет преобразован виджет, и путь к заголовочному файлу:



При таком описании классов рисование кривых будет выглядеть следующим образом:

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: <http://forum.codenet.ru/q26577/eval+%D0%B2+%D0%A1%23>;

<http://www.codeproject.com/Articles/11939/Evaluate-C-Code-Eval-Function>;

<http://www.beyondlinux.com/2011/08/07/3-method-to-evaluate-expressions/>;

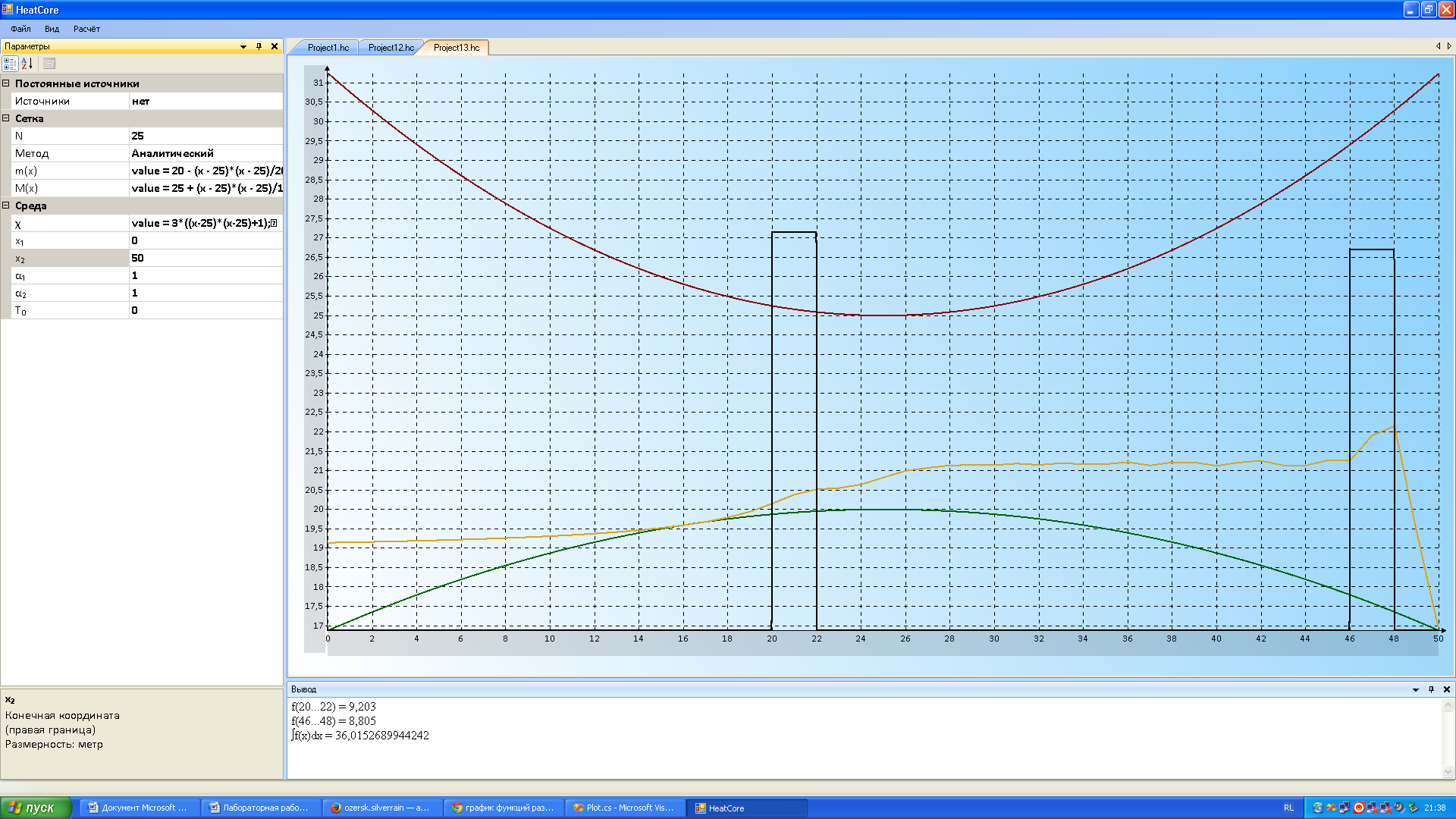
<http://javascript.ru/eval>;

<http://nkoksharov.blogspot.ru/2008/12/rhino-janino.html>.

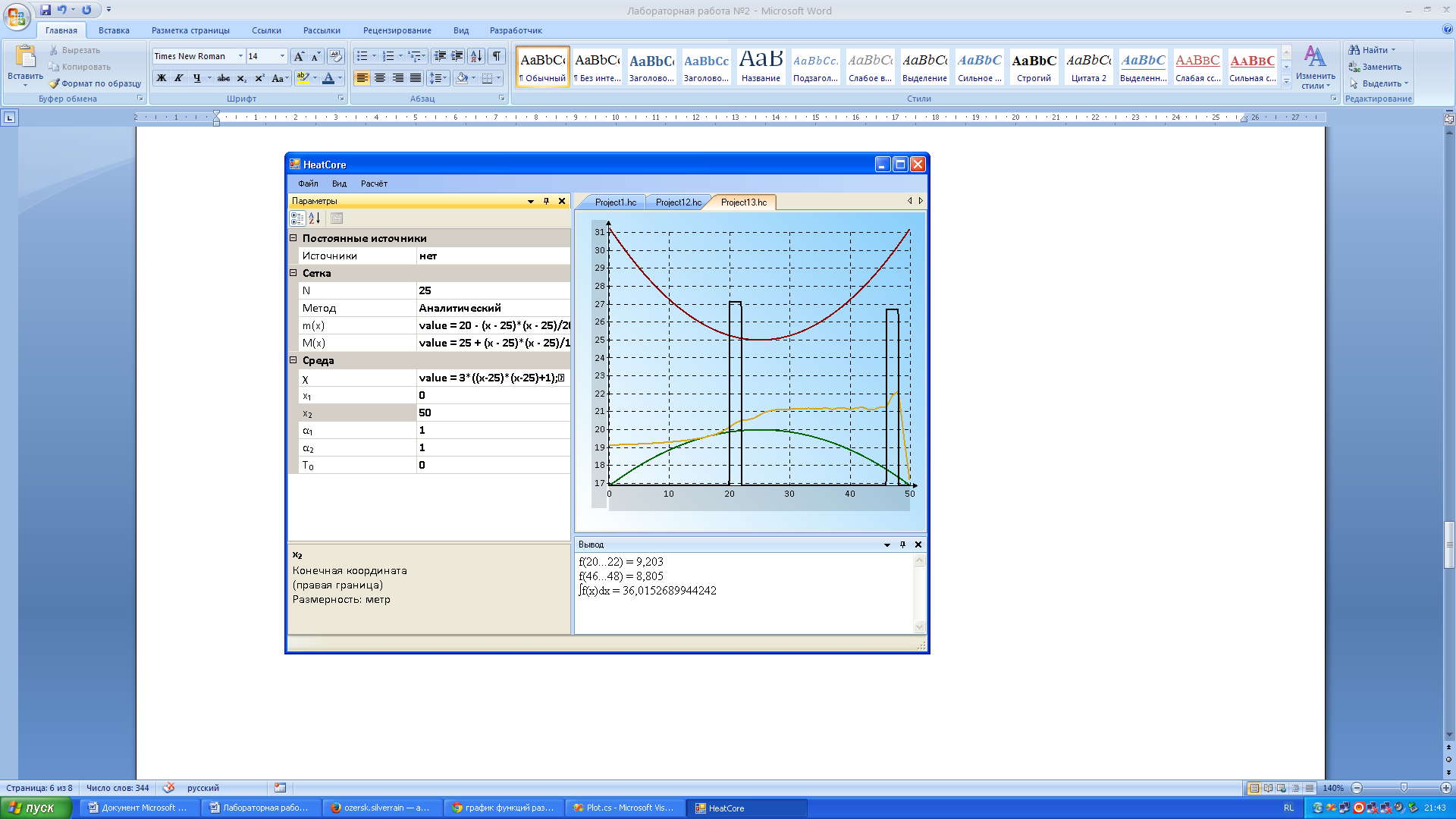
Содержание отчёта

1. Название темы.
2. Цель работы.
3. **Схема структуры классов.**
4. Текст программы.
5. Результат работы программы (снимки экрана).

Пример. Мировой шаг по оси *x* - 2×100, *y* - 5×10-1. Предпочтительный шаг сетки *h* = 30 пикселей.



Поведение того же графика при изменении размера окна (шаг в мировой системе координат увеличивается, экранный остаётся приблизительно тем же):



Пример разметки с маленьким шагом по оси *x*:

