Отчет по Лаборатоорной работе №3

Технология программирования

Бекауов Артур Тимурович

Содержание

# 1 Цель работы

Написать компьютерную программу, содержащую описание классов для иерархии геометрических объектов (точка, линия, квадрат, ромб, прямоугольник, параллелограмм) с реализацией набора методов (изобразить, убрать, передвинуть, повернуть).

# 2 Ход лабораторной работы

**Описание абстрактного класса grobject**

Программа, написанная мной задаёт классы grobject, tline, tsquare, trekt, tromb, tpar.

Первым делом я создал окно вывода размером 1000 на 1000 пикселей, с заголовком SFML(Рис. [1](#fig:1)). Я сделал это до описания всех классов, потому что почти все они используют окно в функции draw.



Figure 1: Создание окна вывода

Затем я описал grobject - базовый, абстрактный класс (Рис. [2](#fig:2)), поэтому экземпляров у него нет. Описывается четырьмя полями - начальная координата x int x, начальная координата y int y, цвет int color и видимость int visible.

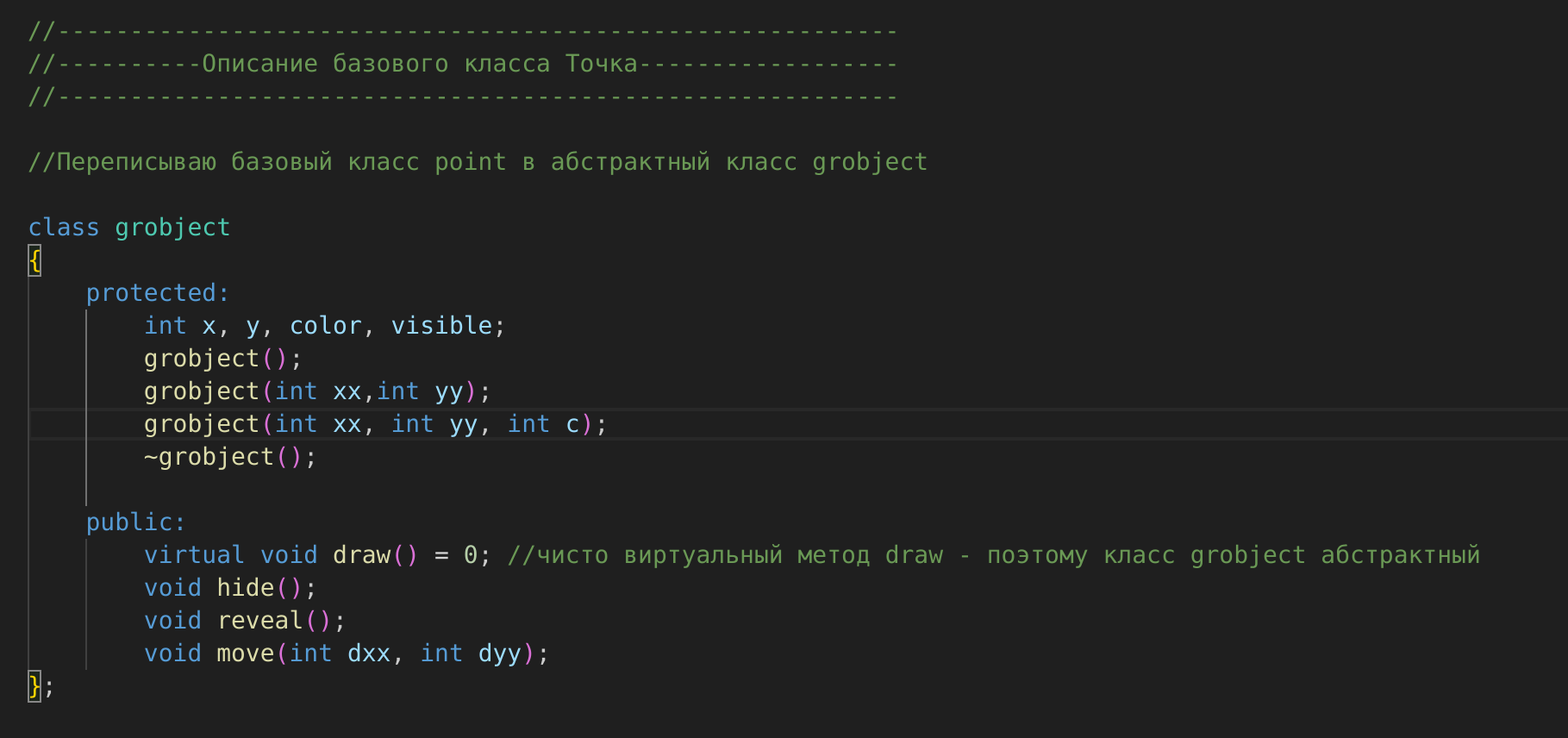


Figure 2: grobject: Описание класса

По существу класс grobject задаёт начальную точку цвет и видимость - параметры, которые унаследуют остальные классы.

Далее прописаны конструкторы и деструктор, вообще для абстрактного класса это не обязательно, но так как я переписывал абстрактный класс grobject из обычного класса point, у меня уже были конструкторы и деструктор, я решил их просто не убирать. Конструкторы и деструктор:

-grobject(); - конструктор по умолчанию (x=500, y=500, color=1, visible=1).

-grobject(int xx, int yy); - конструктор c заданными x, y (x=xx, y=yy, color=1,visible=1).

-grobject(int xx, int yy, int c); - конструктор c заданными x, y, color (x=xx, y=yy, color=c, visible=1).

-~grobject(); - деструктор.

После этого перечисленны методы абстрактного класса grobject (Рис. [3](#fig:3)):

-virtual void draw() = 0; - чисто виртуальный метод draw - именно его наличие определяет класс grobject как абстрактный. В производных классах её придётся переопределять, в отличии от трёъ других функций, которые будут унаследованы всеми остальными классами.

-void hide(); - метод, который отключает видимость объекта, т.е он не будет отрисовываться на экране.

-void reveal(); - метод, который включает видимость объекта. т.е он будет отображаться на экране.

void move(dxx, dyy); - метод, который отклоняет координату начальной точки на dxx и dyy.

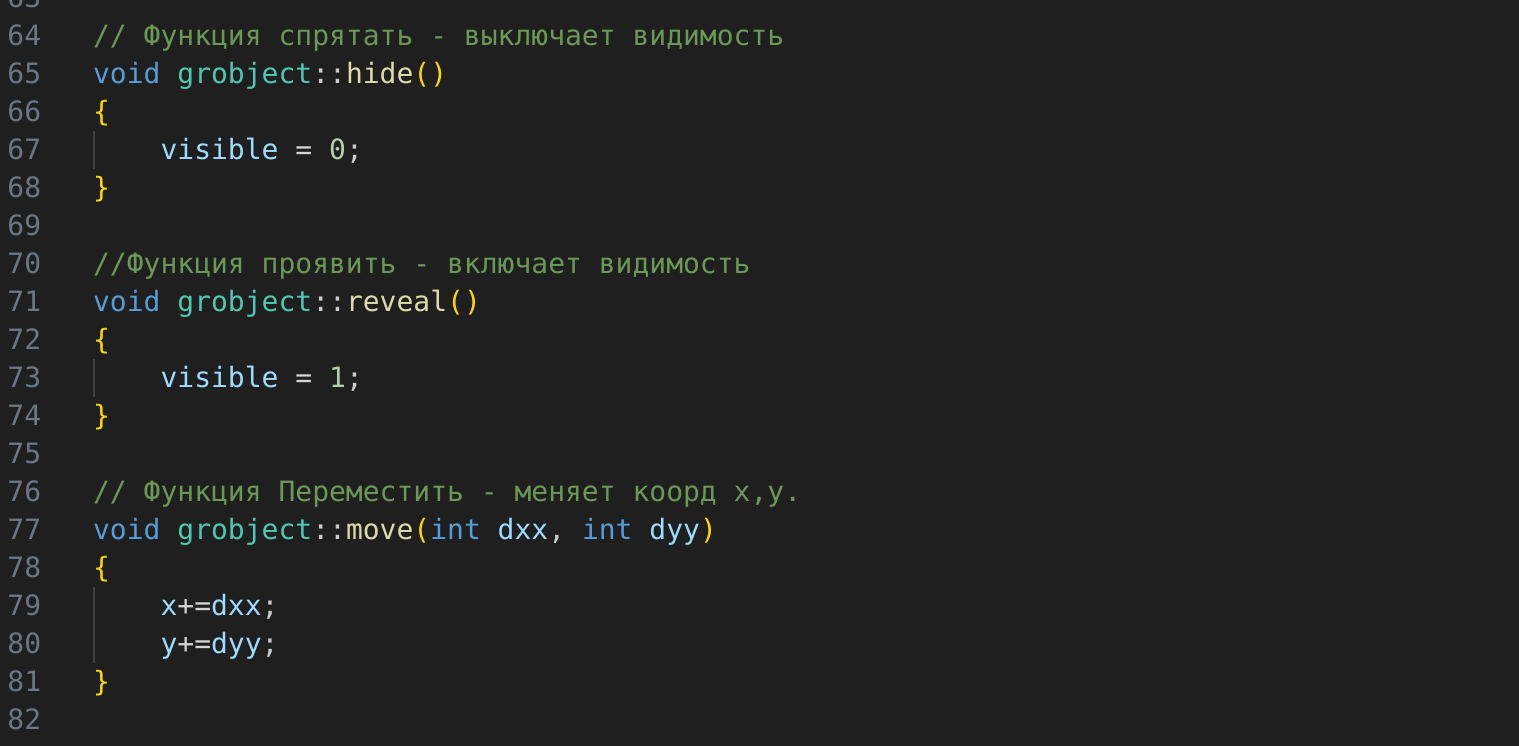


Figure 3: grobject: Методы

**Описание производного (от grobject) класса tline**

После этого я описал класс tline (производный от класса grobject) (Рис. [4](#fig:4)), экземплярами которого являются прямые линии. Помимо полей описывающих начальную точку, видимость и цвет, унаследованных от класса grobject, класс tline имеет поля задающие конечную точку линии: отклонение конечной точки от начальной по координате x - int dx и отклонение конечной точки от начальной по координате y - int dy.

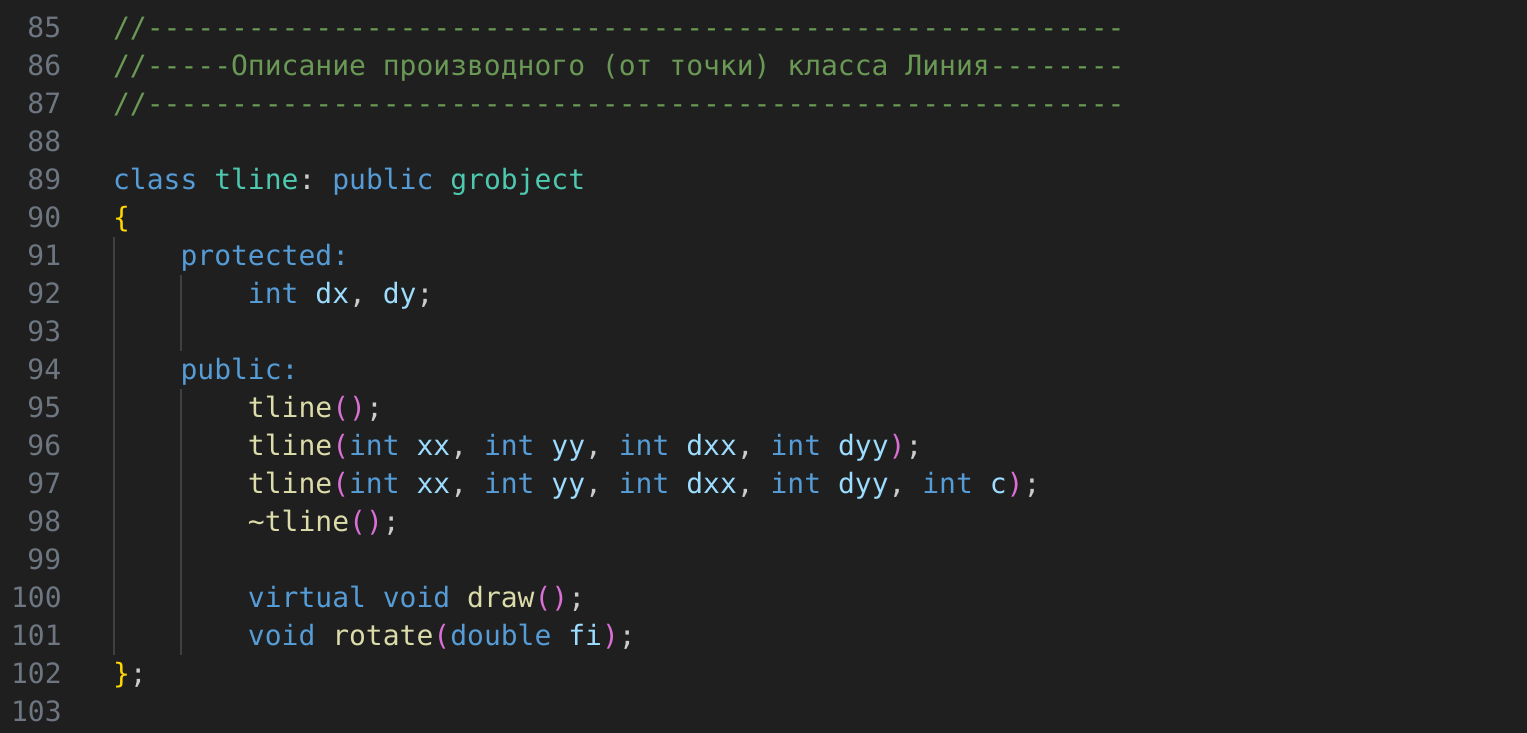


Figure 4: tline: Описание класса

Далее прописаны конструкторы и деструктор класса tline

-tline(); - конструктор по умолчанию.

-tline(int xx, int yy, int dxx, int dyy); - конструктор с заданными параметрами x, y, dx, dy.

-tline(int xx, int yy, int dxx, int dyy, int c); - конструктор с заданными параметрами и цветом.

-~tline(); - деструктор.

Затем я прописал методы класса tline(Рис. [5](#fig:5) и Рис. [6](#fig:6)). Помимо унаследованных от grobject методов hide, reveal и move, также были добавлены:

-virtual void draw(); - чисто виртуальный метод draw класса grobject был переопределён как виртуальный (фактический) метод draw класса tline. Если visible = 1, метод рисует линию в созданном в начале окне. Цвет линии определяется полем color (1 - красный, 2 - зелёный, 3 - синий, 4 - жёлтый, 5 - чёрный, ост - белый). (В этом классе и далее метод draw переопределяется как виртуальный. Сделано это для реализации механизма позднего связывания.)

-void rotate(double fi); - метод rotate. Вращает объект вокруг начальной (x,y) по часовой стрелке на угол fi double fi. Угол fi указывается в градусах.



Figure 5: tline: Методы (1\2)

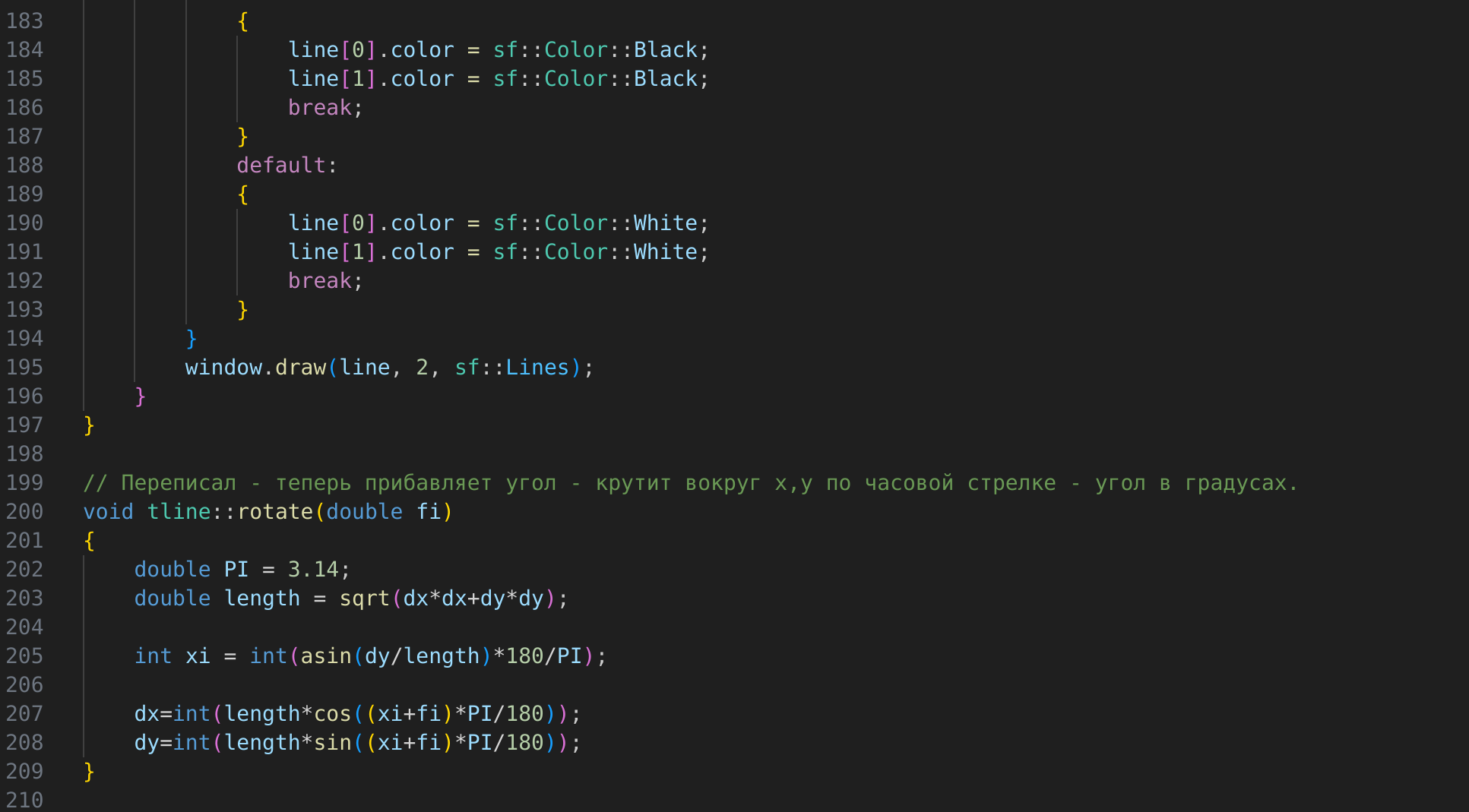


Figure 6: tline: Методы (2\2)

**Описание производного (от tline) класса tsquare**

После этого я описал класс tsquare (производный от класса tline) (Рис. [7](#fig:7)), экземплярами которого являются квадраты. Помимо полей описывающих начальную вершину, видимость, цвет и отклонение второй вершины от начальной, унаследованных от tline, класс tsquare не имее уникальных полей.

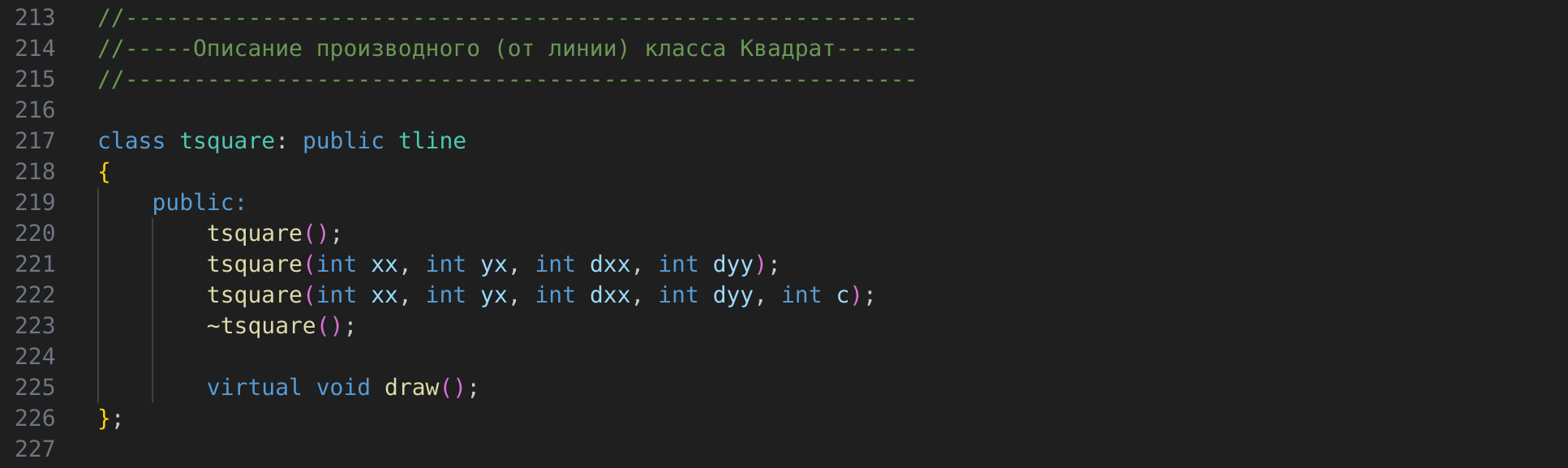


Figure 7: tsquare: Описание класса

Далее прописаны конструкторы и деструктор класса tsquare:

-tsquare(); - конструктор по умолчанию.

-tsquare(int xx, int yy, int dxx, int dyy); - конструктор с заданными параметрами x, y, dx, dy.

-tsquare(int xx, int yy, int dxx, int dyy, int c); - конструктор с заданными параметрами и цветом.

-~tsquare(); - деструктор.

Затем я прописал методы класса tsquare (Рис. [8](#fig:8) и Рис. [9](#fig:9)). Помимо унаследованных от tline методов hide, reveal, move, и rotate, также были добавлены:

–virtual void draw(); - был переопределён виртуальный метод draw. Если visible = 1, метод рисует квадрат в созданном в начале окне. Цвет квадрата определяется полем color.

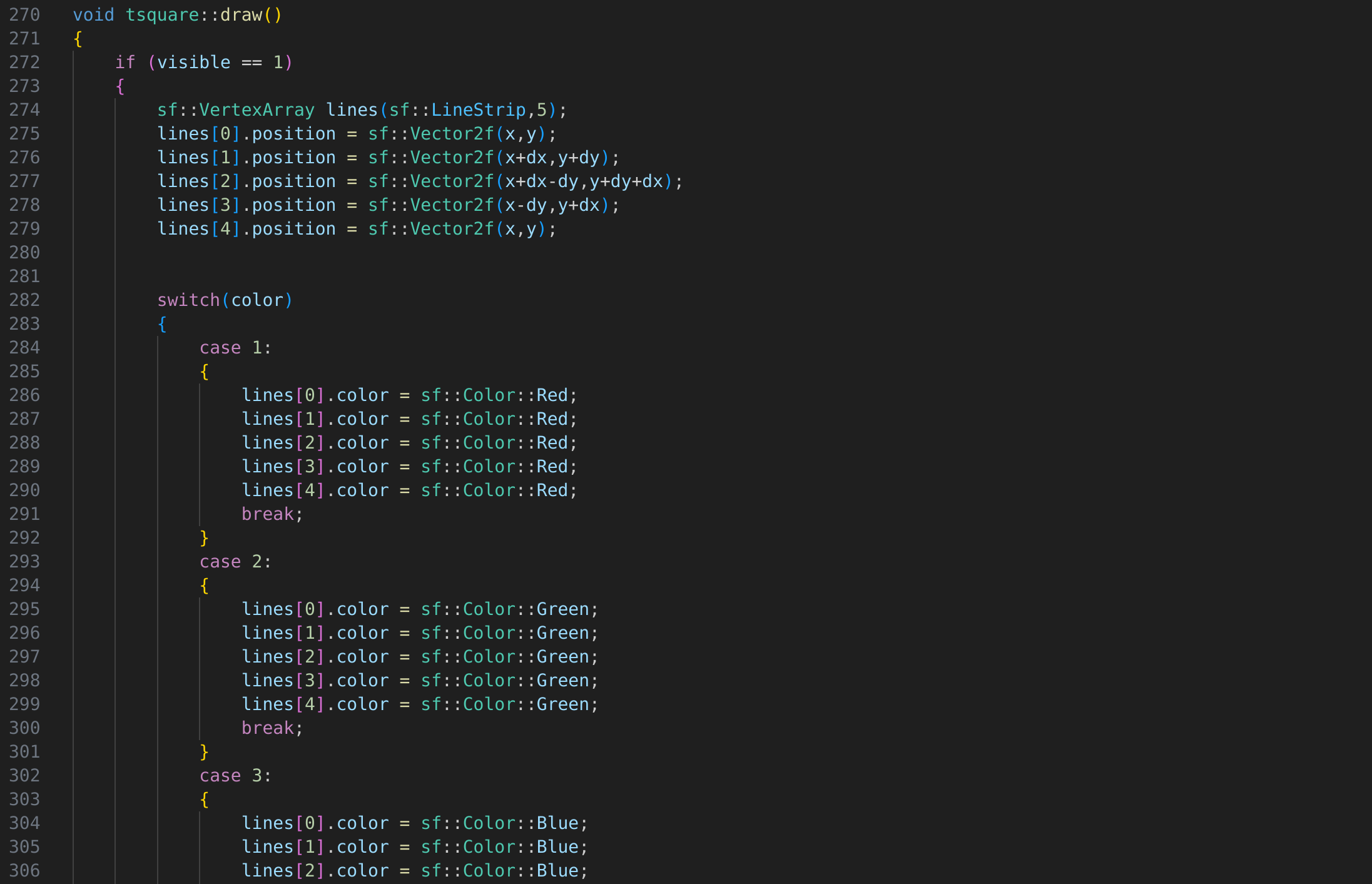


Figure 8: tsquare: Методы (1\2)

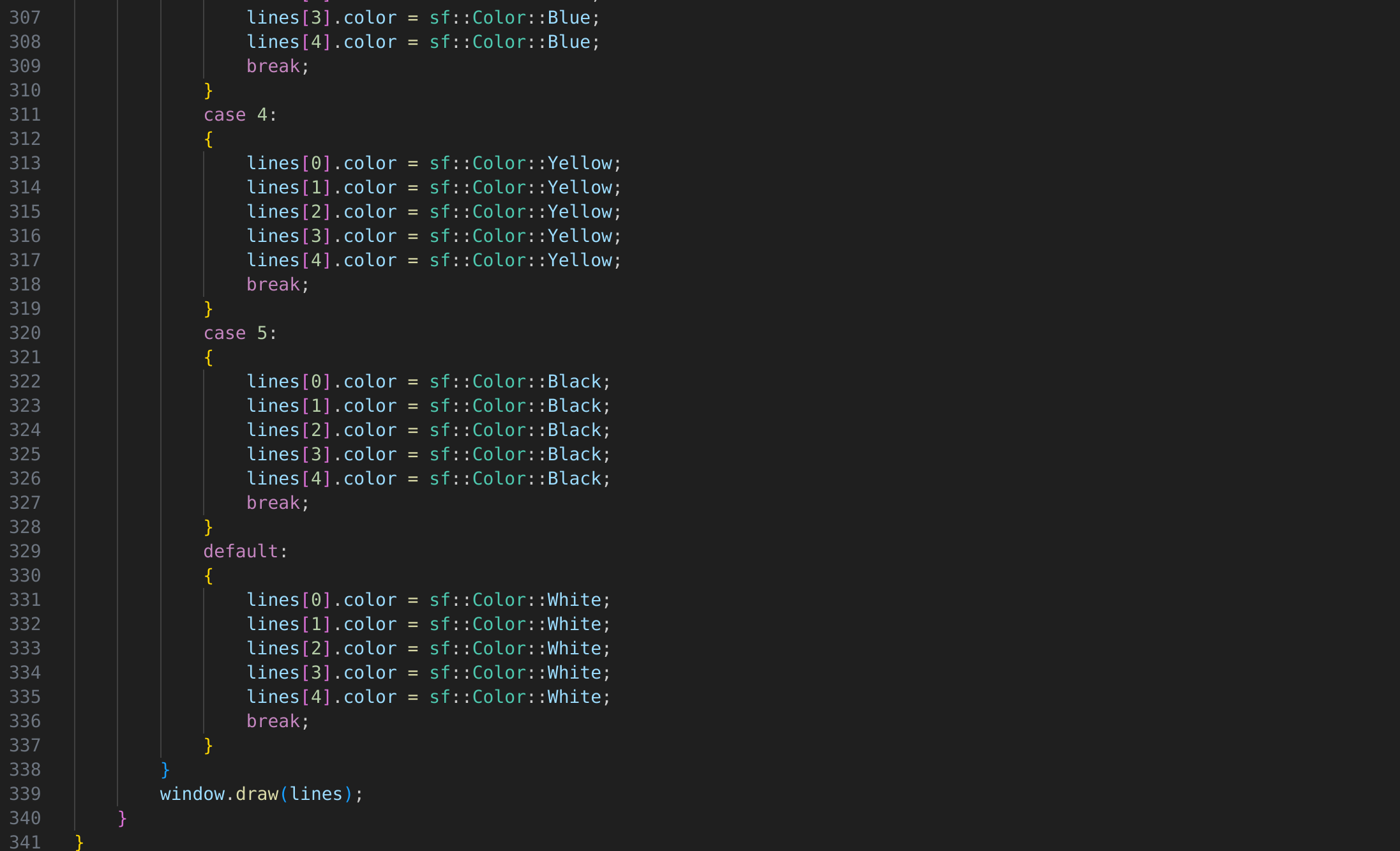


Figure 9: tsquare: Методы (2\2)

**Описание производного (от tsquare) класса trekt**

После этого я описал класс trekt (производный от виртуального класса tsquare)(Рис. [10](#fig:10))(Класс tsquare в наследовании описан как виртуальный, равно как и у tromb. Сделано это потому что класс tpar будет иметь два предка (tromb и trekt), и унаследует два комплекта одинаковых полей tsquare от них. Чтобы избежать путанницы с уканием к какому полю идёт обращение, класс предок объявляется виртуальным, у trekt и tromb). Экземплярами класса являются прямоугольники. Помимо полей описывающих начальную вершину, видимость, цвет и отклонение второй вершины от начальной, унаследованных от tsquare, класс trekt имеет поля коэфициентов увеличения первой стороны float ak и второй стороны float bk. Первой стороной считается сторона между вершинами (x,y) и (x+dx,y+dy). Если ak и bk равны 1, то получается обычный квадрат с теми же параметрами.

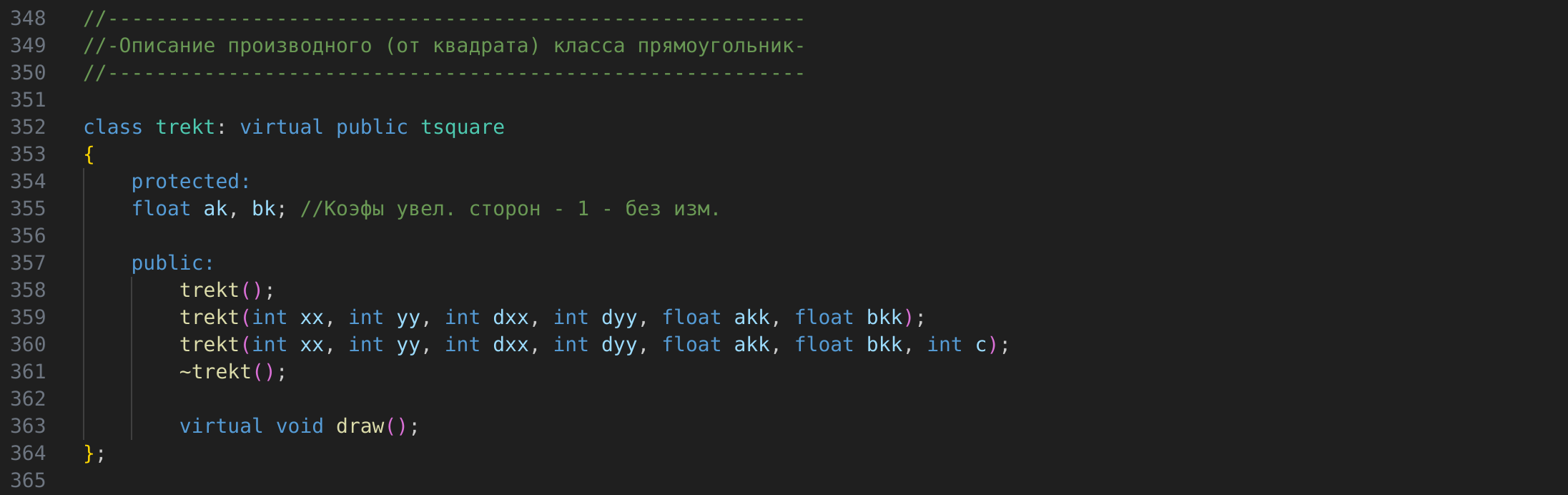


Figure 10: trekt: Описание класса

Далее прописаны конструкторы и деструктор класса trekt:

-trekt(); - конструктор по умолчанию.

-trekt(int xx, int yy, int dxx, int dyy, float akk, float bkk); - конструктор с заданными параметрами x, y, dx, dy, ak, bk.

-trekt(int xx, int yy, int dxx, int dyy, float akk, float bkk, int c); - конструктор с заданными параметрами и цветом.

-~trekt(); - деструктор.

Затем я прописал методы класса trekt (Рис. [11](#fig:11) и Рис. [12](#fig:12)). Помимо унаследованных от tsquare методов hide, reveal, move, и rotate, также были добавлены:

–virtual void draw(); - был переопределён виртуальный метод draw. Если visible = 1, метод рисует прямоугольник в созданном в начале окне. Цвет прямоугольника определяется полем color.

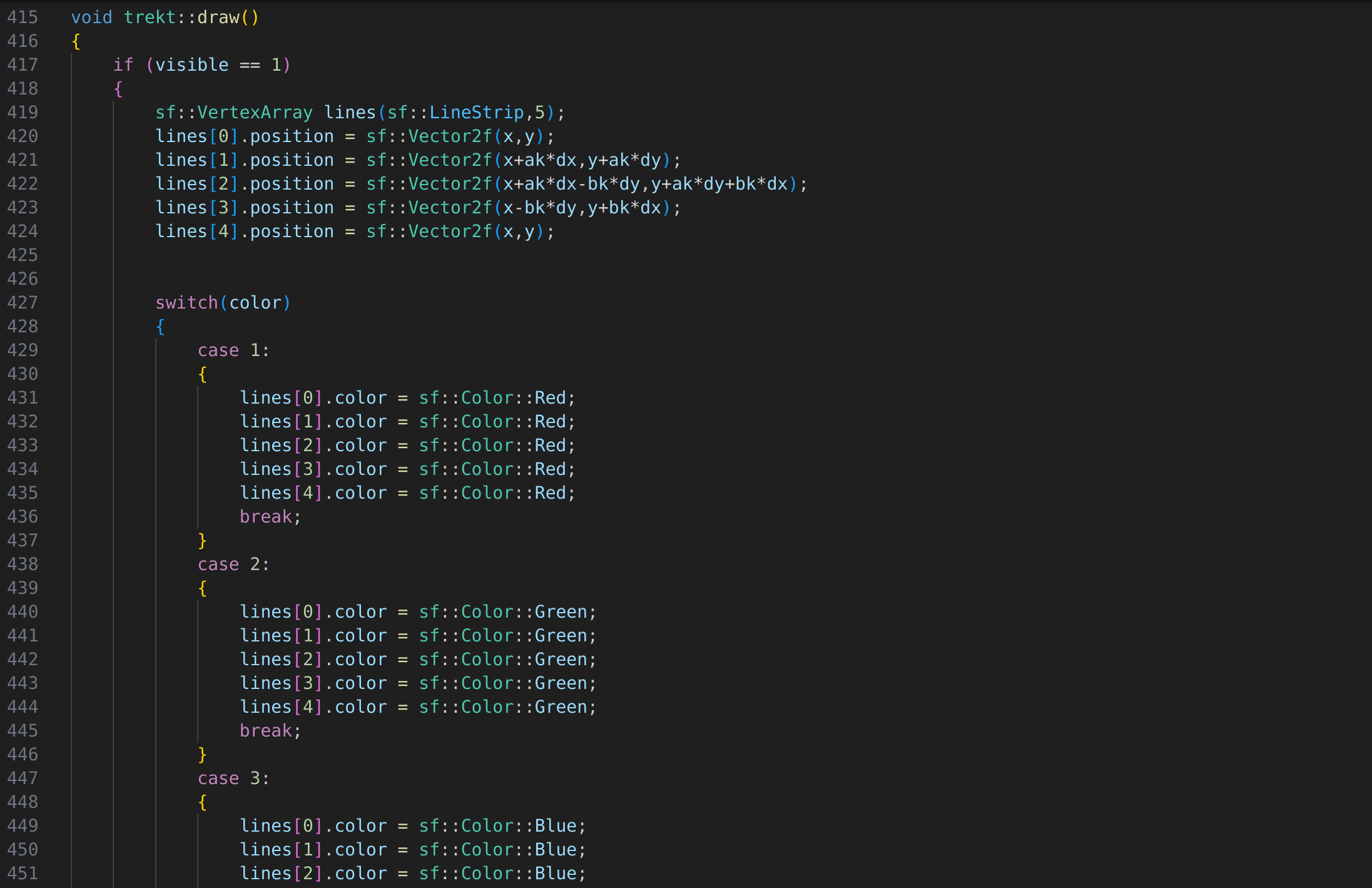


Figure 11: trekt: Методы (1\2)

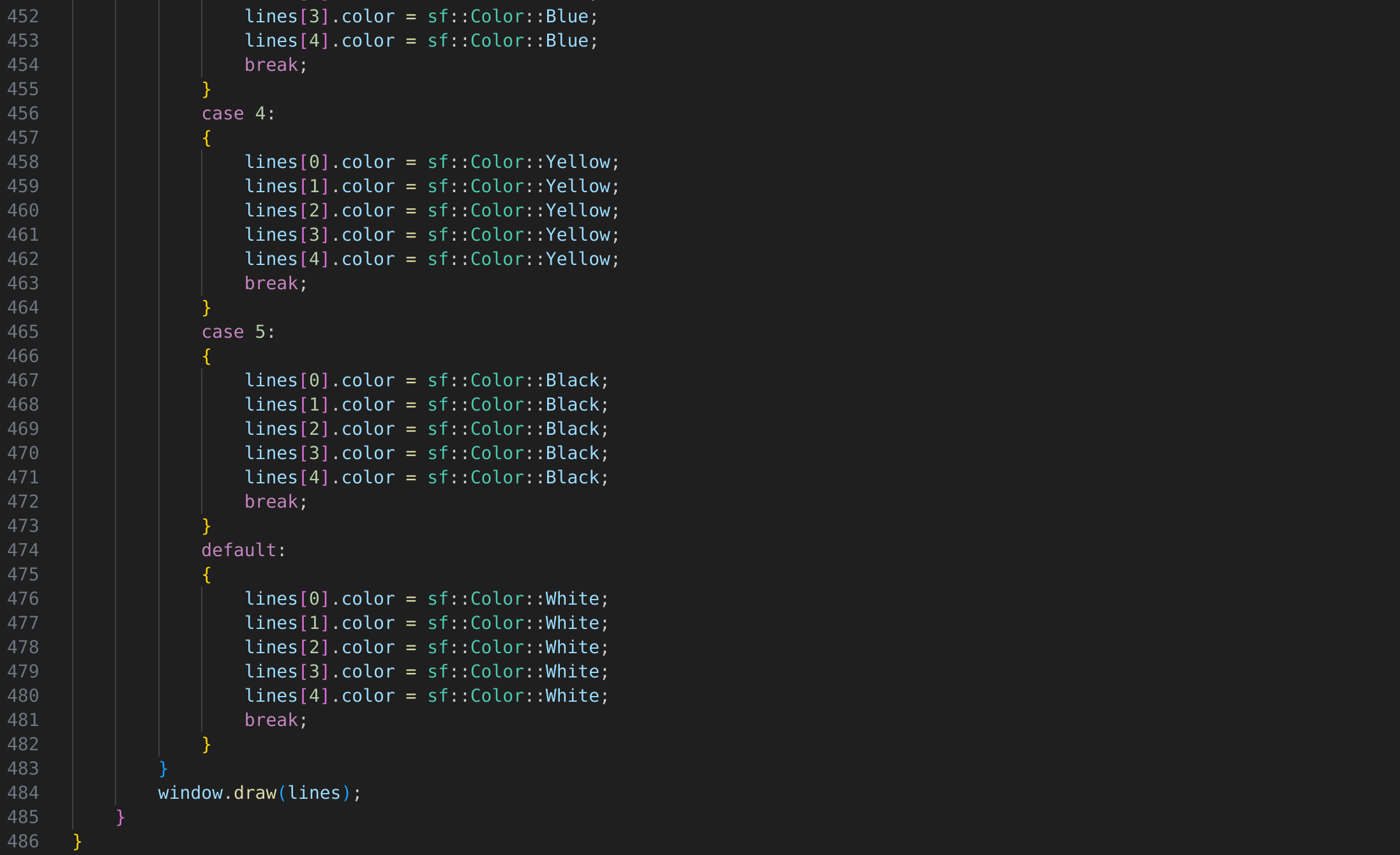


Figure 12: trekt: Методы (2\2

**Описание производного (от tsquare) класса tromb**

После этого я описал класс tromb (производный от виртульного класса tsquare) (Рис. [13](#fig:13)), экземплярами которого являются ромбы. Помимо полей описывающих начальную вершину, видимость, цвет и отклонение второй вершины от начальной, унаследованных от tsquare, класс tromb имеет поле определяющее угол при второй вершине (x+dx,y+dy) int fi. Значение угла следует указывать в градусах, желательно от 0 до 180.

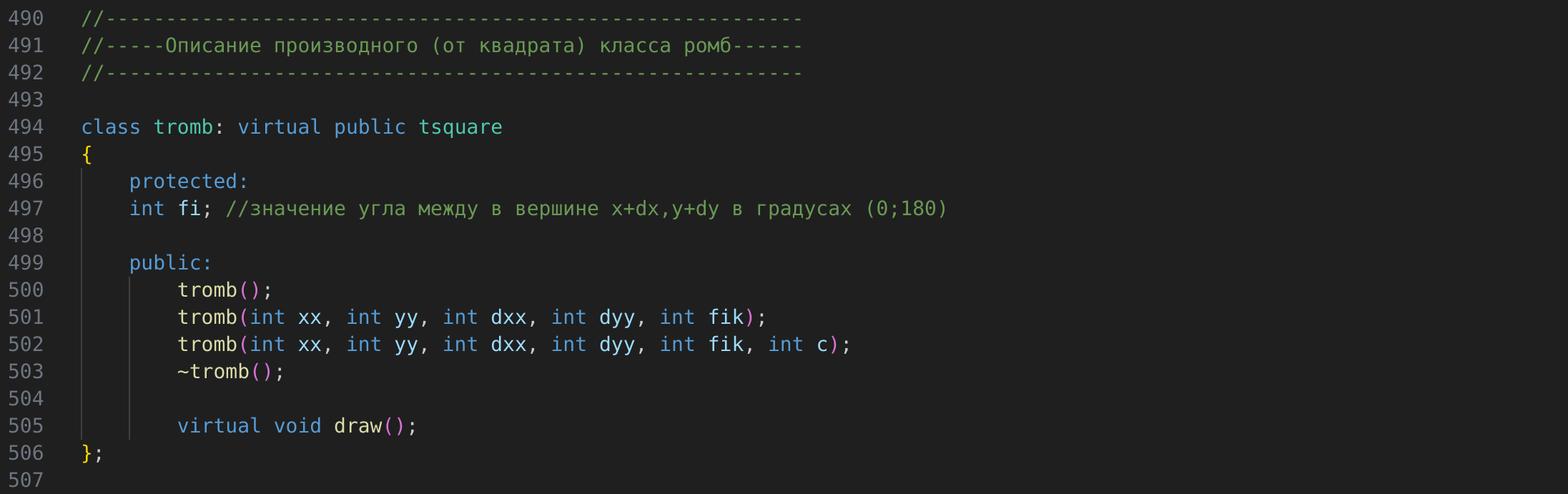


Figure 13: tromb: Описание класса

Далее прописаны конструкторы и деструктор класса tromb:

-tromb(); - конструктор по умолчанию.

-tromb(int xx, int yy, int dxx, int dyy, int fik); - конструктор с заданными параметрами x, y, dx, dy, fi.

-tromb(int xx, int yy, int dxx, int dyy, int fik, int c); - конструктор с заданными параметрами и цветом.

-~tromb(); - деструктор.

Затем я прописал методы класса tromb (Рис. [14](#fig:14) и Рис. [15](#fig:15) и Рис. [16](#fig:16)). Помимо унаследованных от tsquare методов hide, reveal, move, и rotate, также были добавлены:

–virtual void draw(); - был переопределён виртуальный метод draw. Если visible = 1, метод рисует ромб в созданном в начале окне. Цвет прямоугольника определяется полем color.

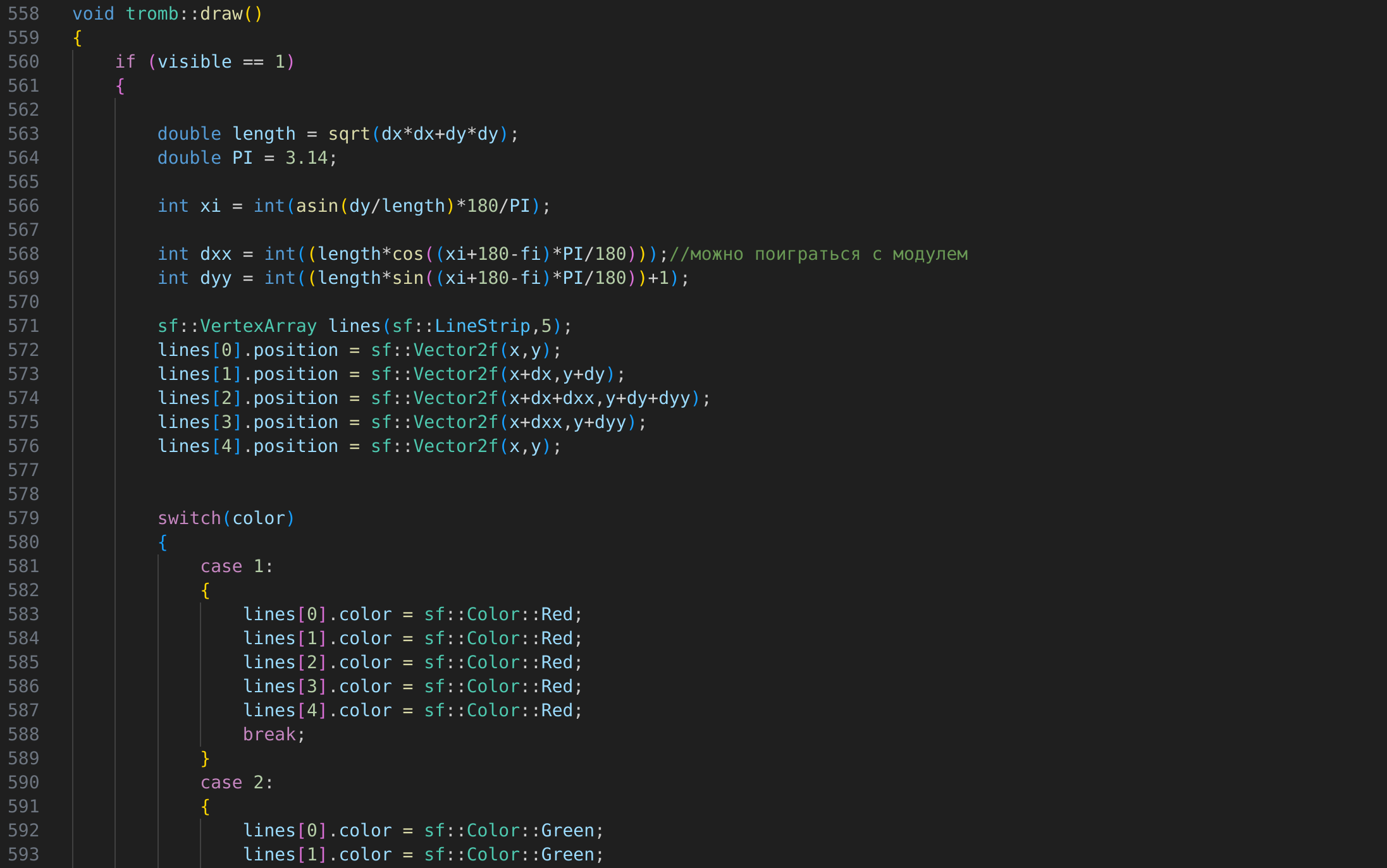


Figure 14: tromb: Методы (1\3)

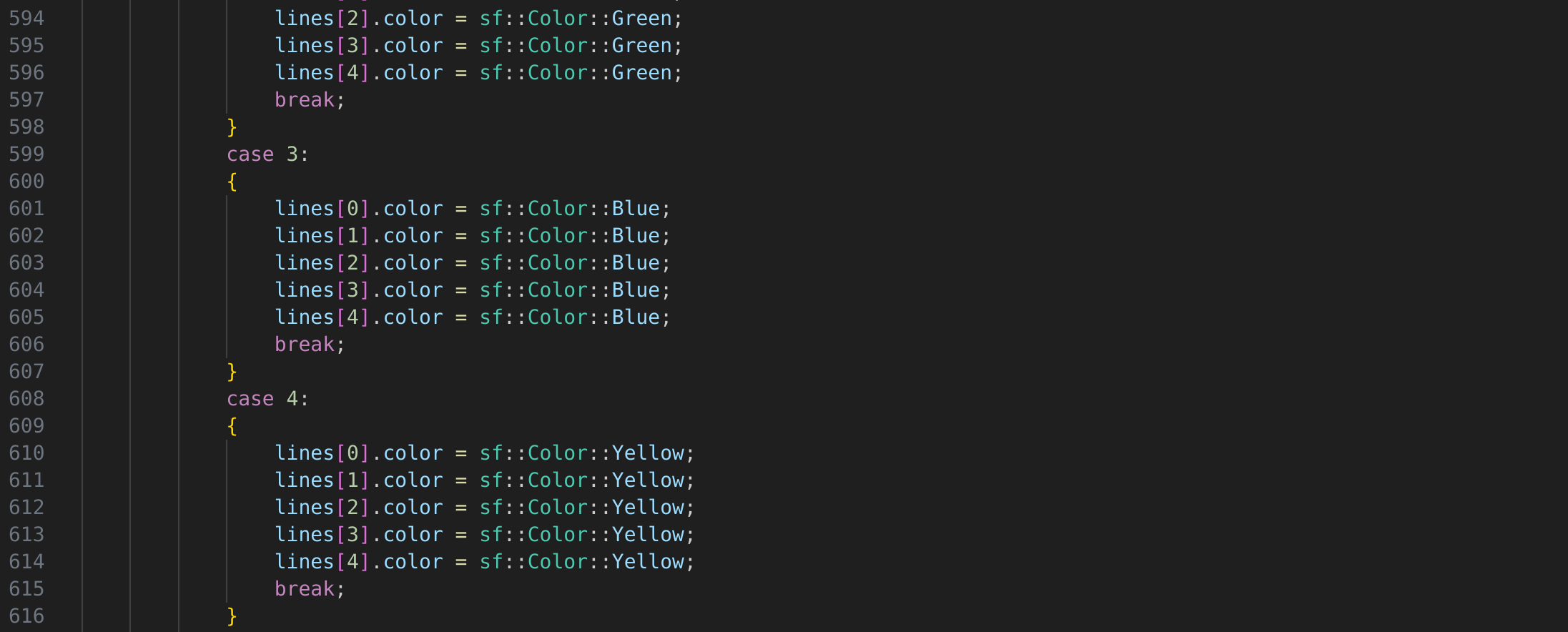


Figure 15: tromb: Методы (2\3)



Figure 16: tromb: Методы (3\3)

**Описание производного (от trekt и tromb) класса tpar**

После этого я описал класс tpar (производный от классов tromb и trekt) (Рис. [17](#fig:17)), экземплярами которого являются параллелограмы. Помимо полей описывающих начальную вершину, видимость, цвет , отклонение второй вершины от начальной, унаследованных от tsquare, класс tpar имеет поля коэфициентов увеличения сторон (унаследованные от trekt) и поле определяющее угол при второй вершине (унаследованное у tromb).

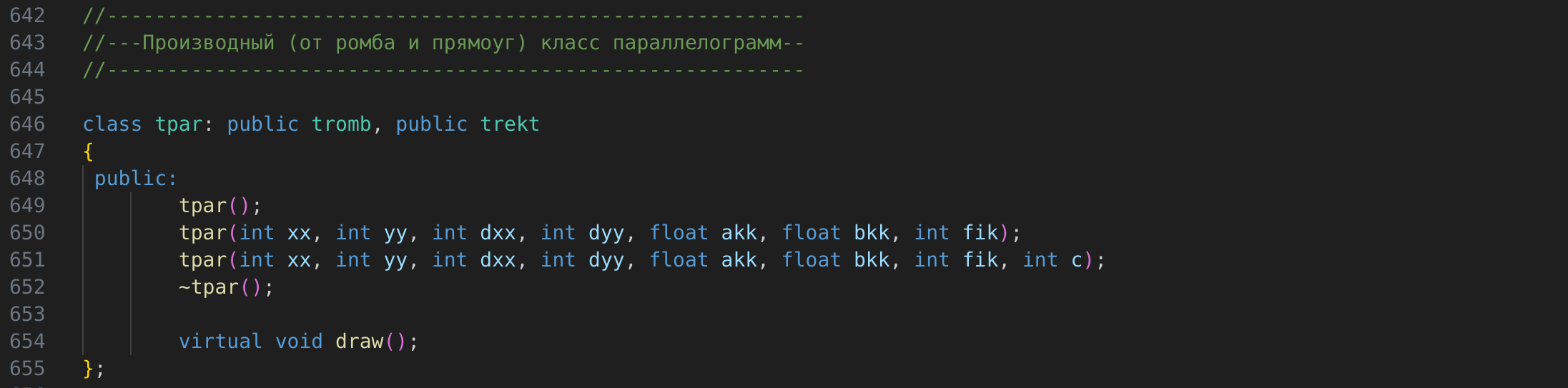


Figure 17: tpar: Описание класса

Далее прописаны конструкторы и деструктор класса tpar:

-tpar(); - конструктор по умолчанию.

-tpar(int xx, int yy, int dxx, int dyy, float akk, float bkk, int fik); - конструктор с заданными параметрами x, y, dx, dy, ak, bk, fi.

-tpar(int xx, int yy, int dxx, int dyy, float akk, float bkk,int fik, int c); - конструктор с заданными параметрами и цветом.

-~tpar(); - деструктор.

Затем я прописал методы класса tpar (Рис. [18](#fig:18) и Рис. [19](#fig:19) и Рис. [20](#fig:20)). Помимо унаследованных от tsquare методов hide, reveal, move, и rotate, также были добавлены:

–virtual void draw(); - был переопределён виртуальный метод draw. Если visible = 1, метод рисует параллелограм в созданном в начале окне. Цвет прямоугольника определяется полем color.

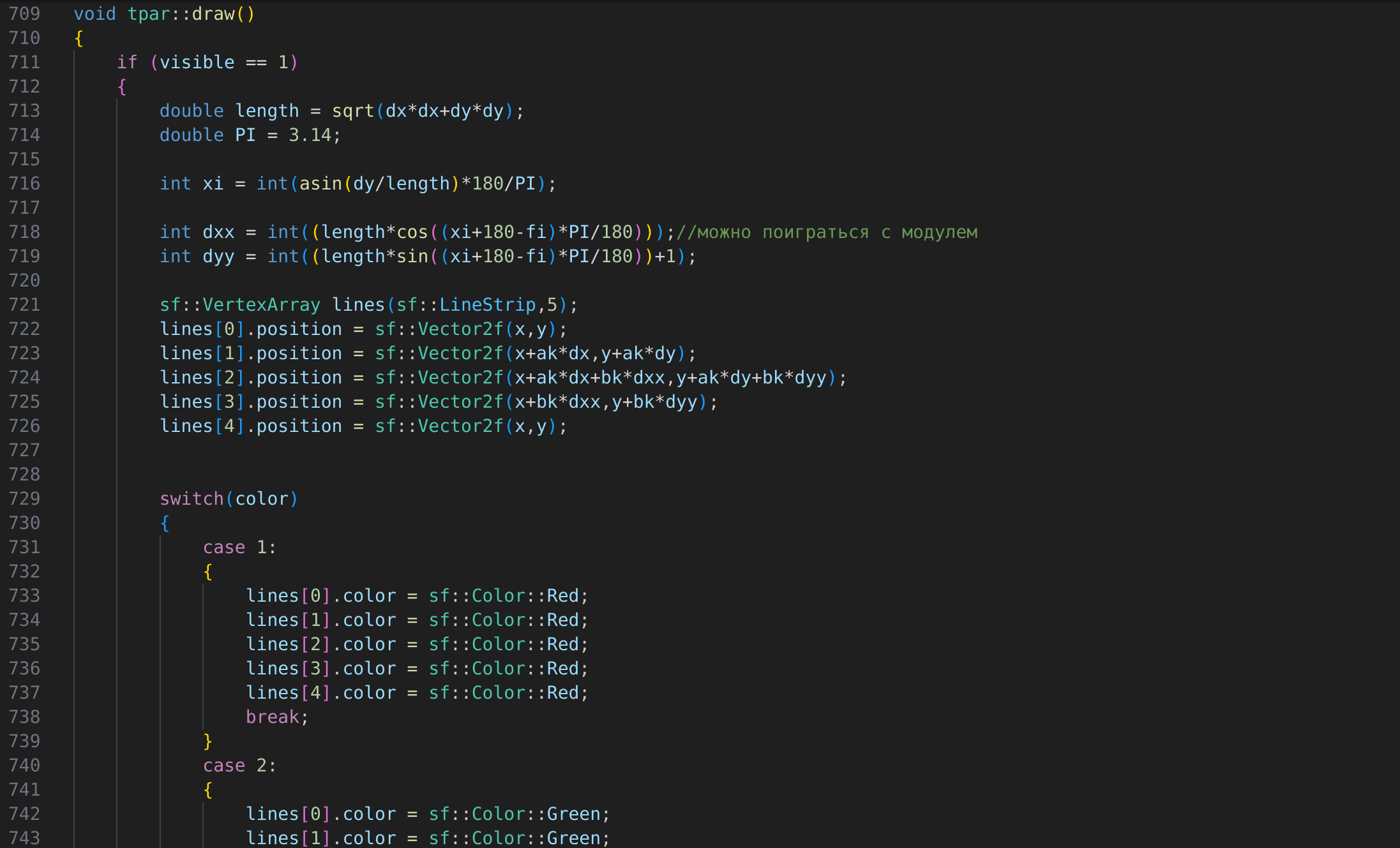


Figure 18: tpar: Методы (1\3)

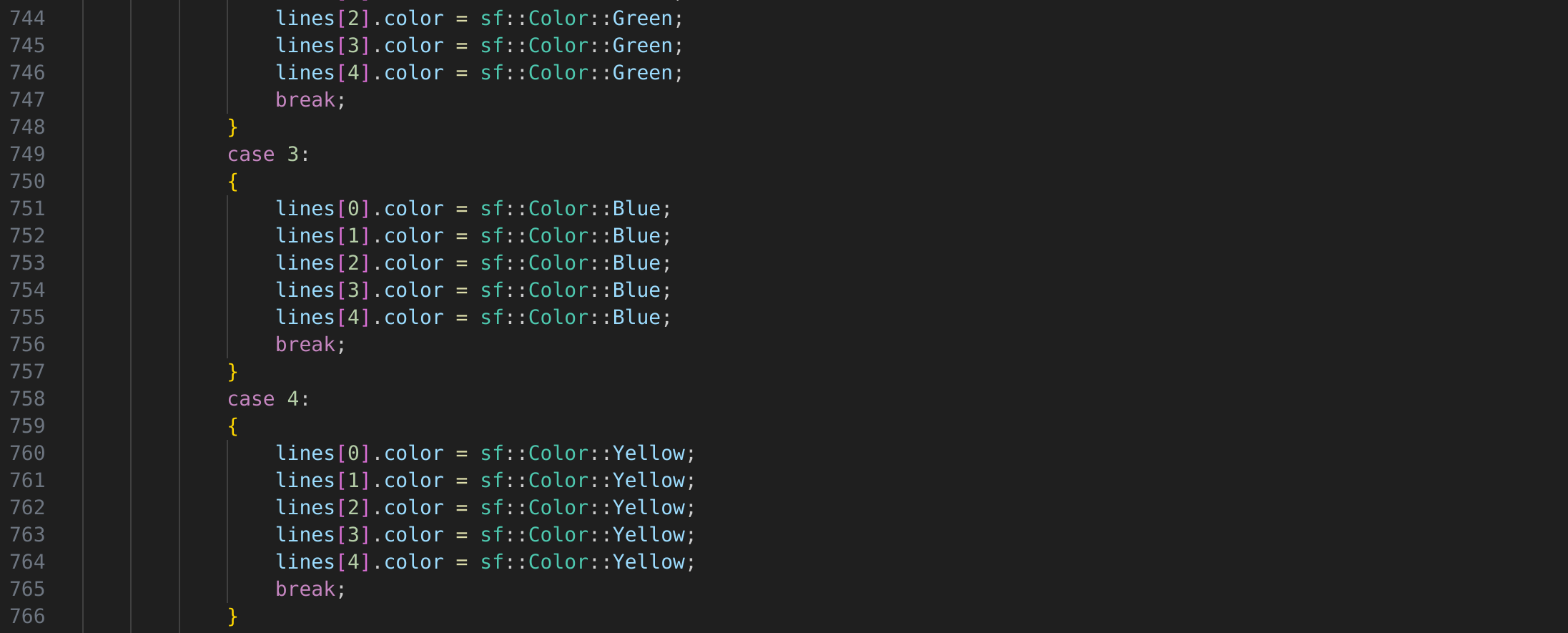


Figure 19: tpar: Методы (2\3)

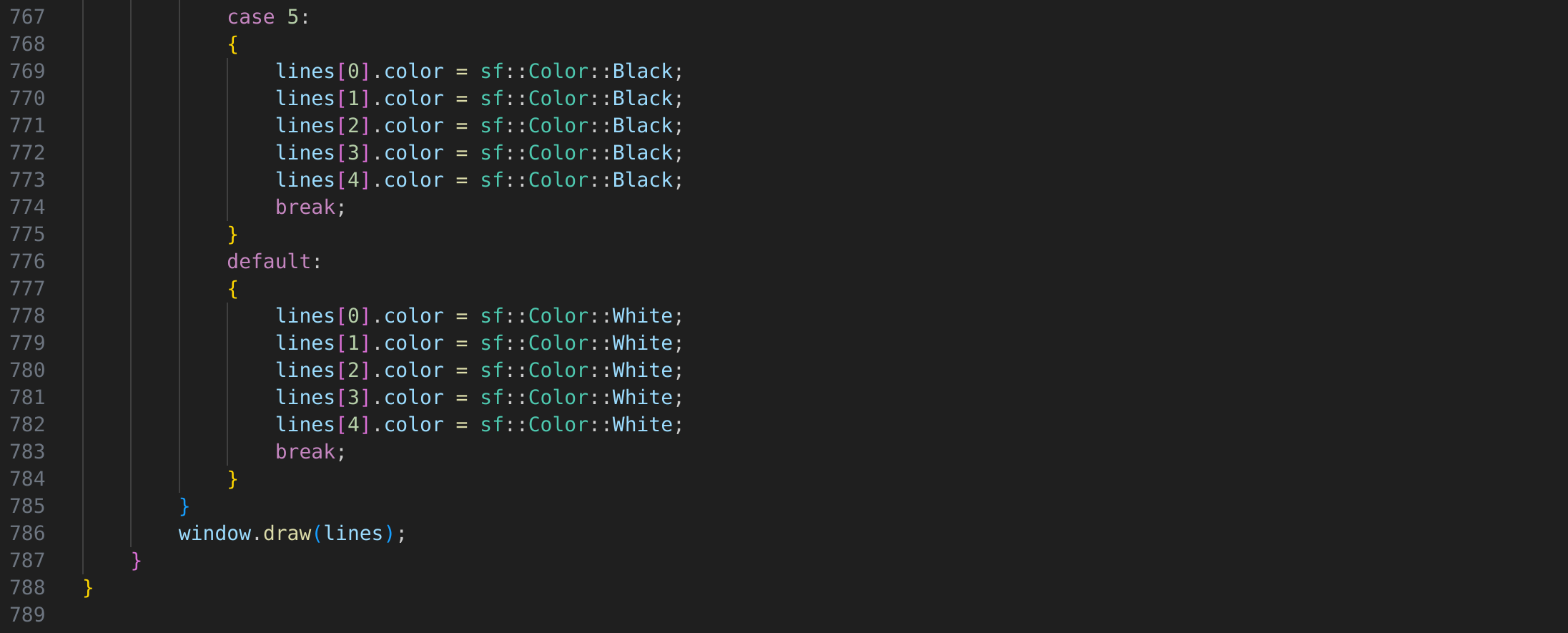


Figure 20: tpar: Методы (3\3)

**Функция int main()**

В функции main я создал белую сетку из экземпляров класса tline, а также добавил по 3 экземпляра классов tsquare, tromb, trekt, tpar, один из которых в каждом классе скрываю функцией hide. (Рис. [21](#fig:21) и Рис [22](#fig:22)).

После этого я описываю указатель на объект класса квадрат tsquare \*fig. Затем я выделяю в динамической памяти объект, являющийся ромбом по умолчанию и присваиваю указателю fig значение этой памяти.

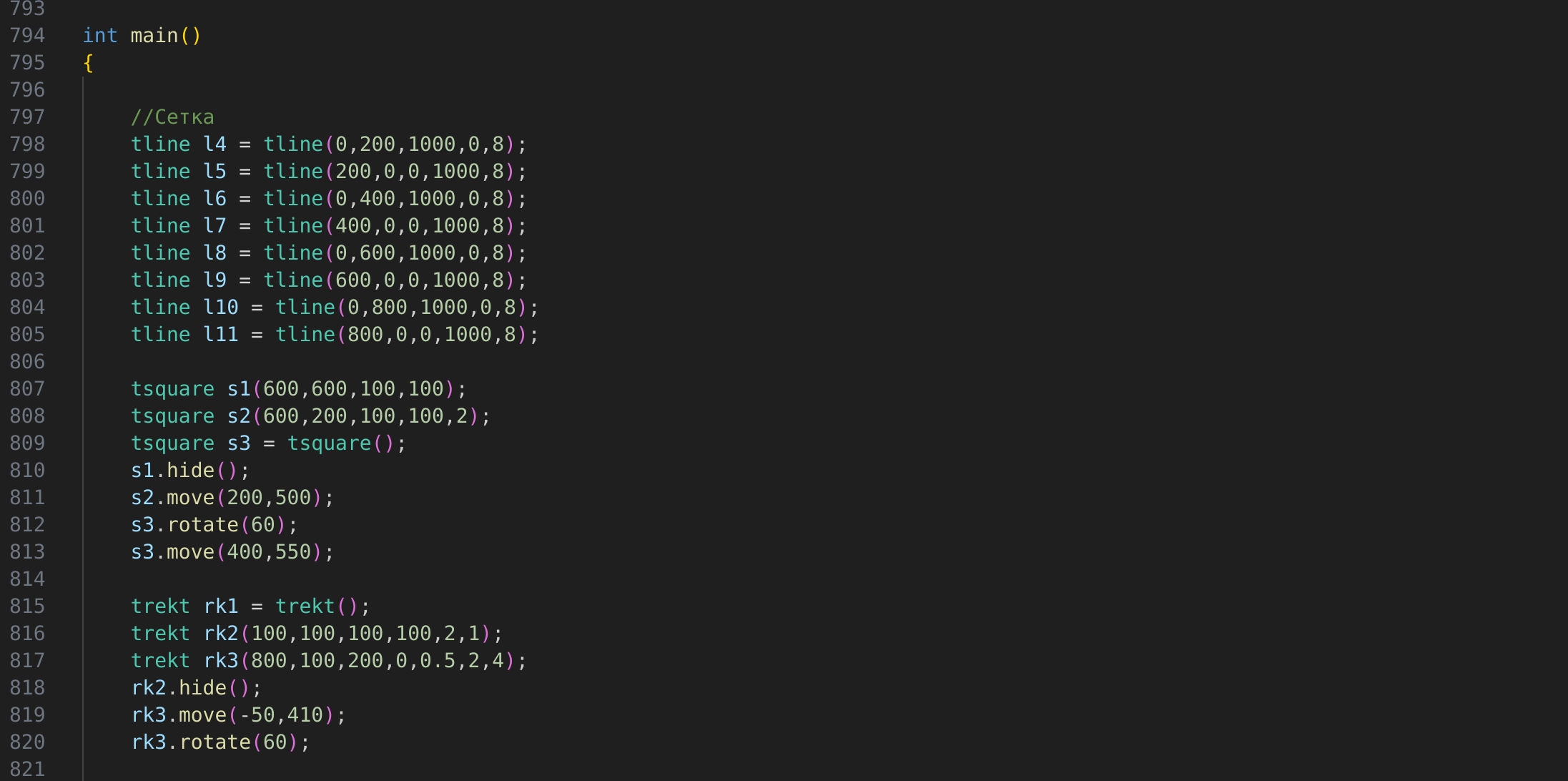


Figure 21: Функция main: Создание экземпляров (1\2)

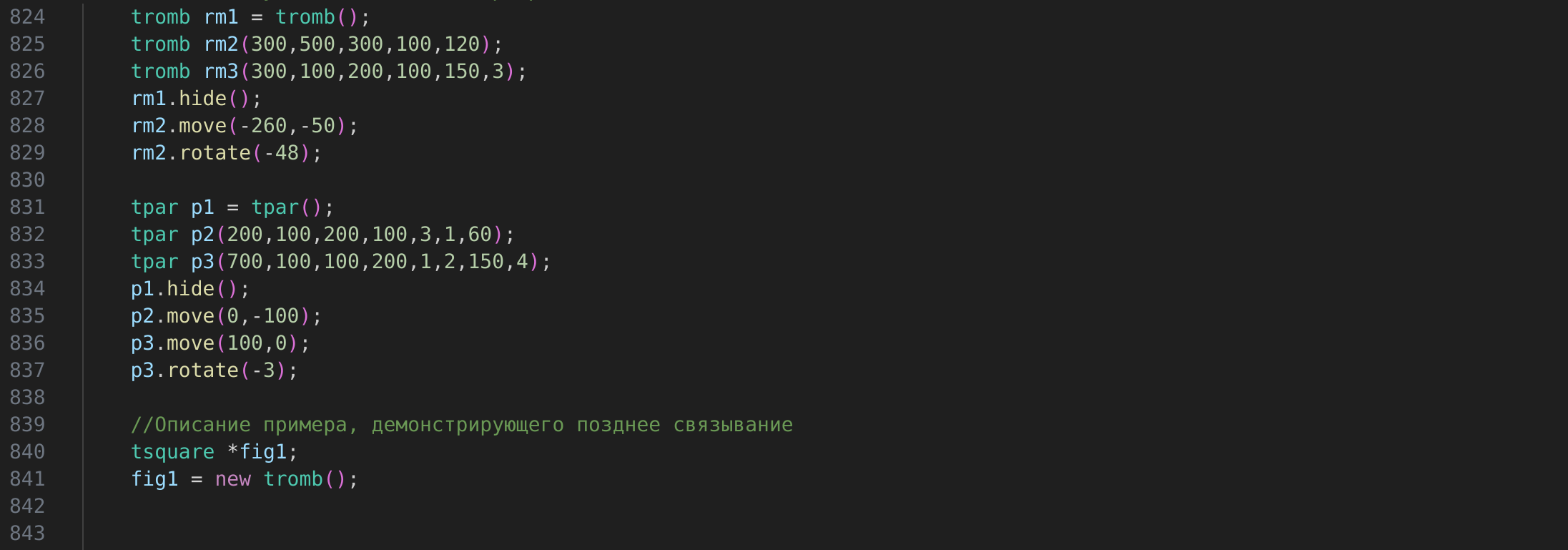


Figure 22: Функция main: Создание экземпляров (2\2)

Далее идёт цикл, который выводит окно вывода на экран, в цикле описаны все выводимые на экран элементы (через функцию draw, ведь именно она отрисовывает объект на экране)(Рис. [23](#fig:23) и Рис [24](#fig:24)).



Figure 23: Функция main: Цикл вывода (1\2)

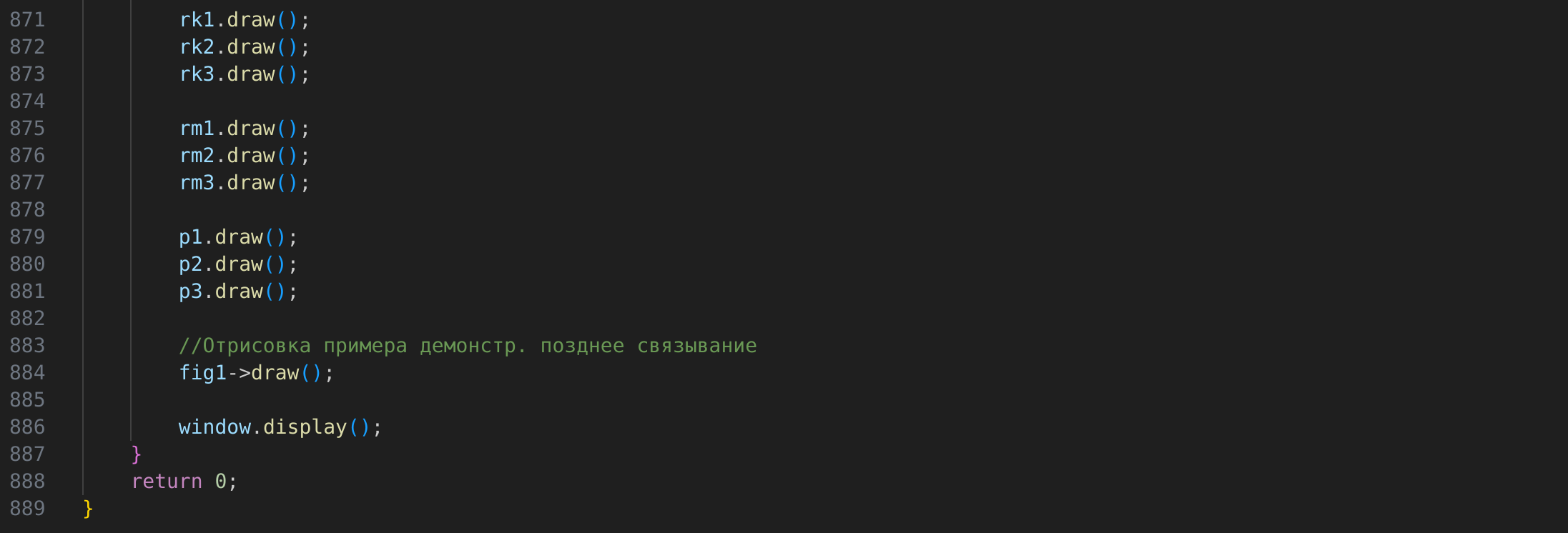


Figure 24: Функция main: Цикл вывода (2\2)

Вывод программы следующий (Рис. [25](#fig:25)):

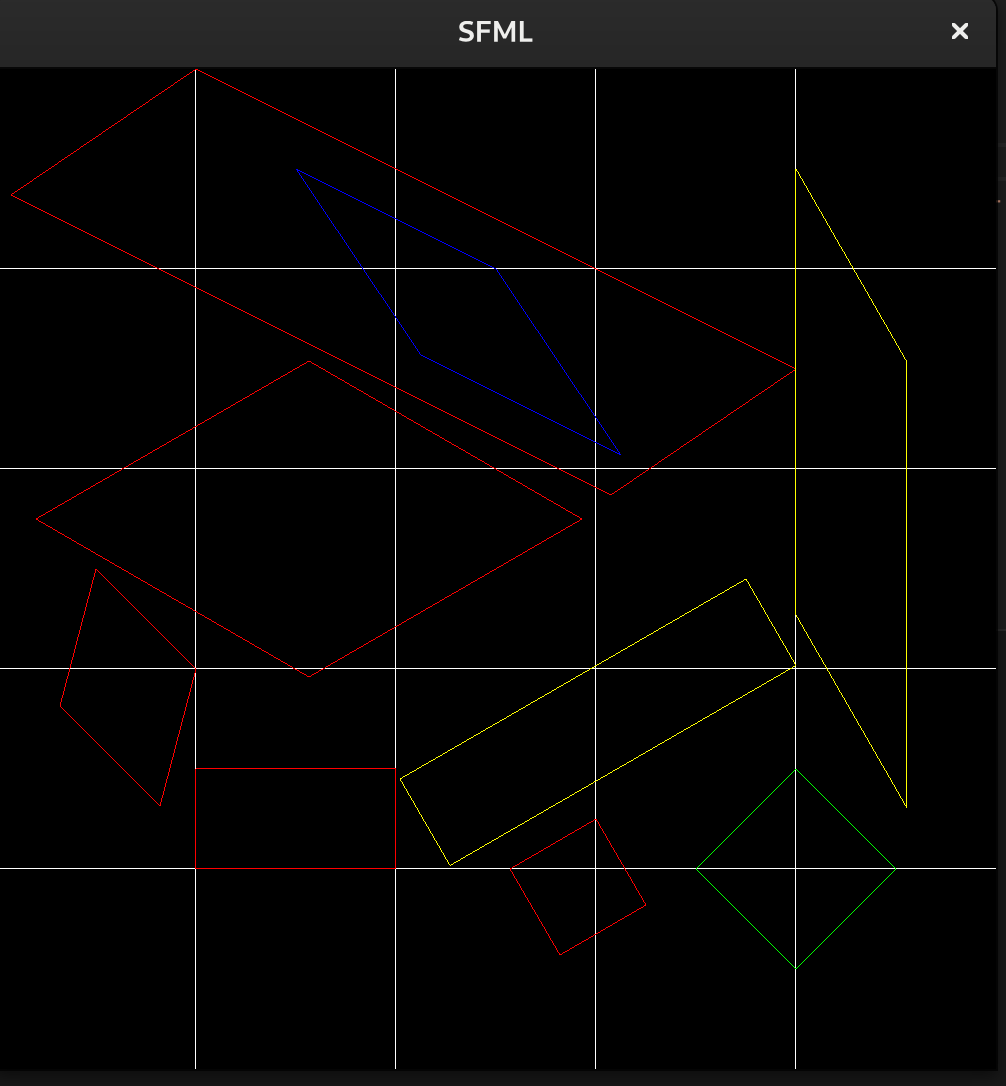


Figure 25: Вывод программы

На рисунке отрисовывается третий ромб, значит, несмотря на то, что fig1 - указатель на экземпляр класса квадрат, на этапе позднего связывания программа понимает, что в динамической памяти находятся данные ромба и применяет соответствующий draw - из производного класса ромба.

# 3 Выводы

В ходе лабораторной работы я написал компьютерную программу на с++, содержащую описание классов для иерархии геометрических объектов (точка, линия, квадрат, ромб, прямоугольник, параллелограмм) с реализацией набора методов (изобразить, убрать, передвинуть, повернуть).