Федеральное агентство связи  
Федеральное государственное бюджетное образовательное учреждение  
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра прикладной математики и кибернетики

Расчетно-графическое задание по дисциплине «Объектно-ориентированное программирование». Задание №13: «Реализовать игру «Тетрис» в текстовом или в графическом режиме».

Выполнил:

студент 2 курса ИВТ

группы ИП – 811

Разумов Д. Б.

Проверила:

Ситняковская Е. И.

**Содержание**

[Введение 3](#_Toc22866309)

[Классы 4](#_Toc22866310)

[Функции 8](#_Toc22866311)

[Алгоритм основной программы 1](#_Toc22866309)0

[Cкриншоты 1](#_Toc22866309)4

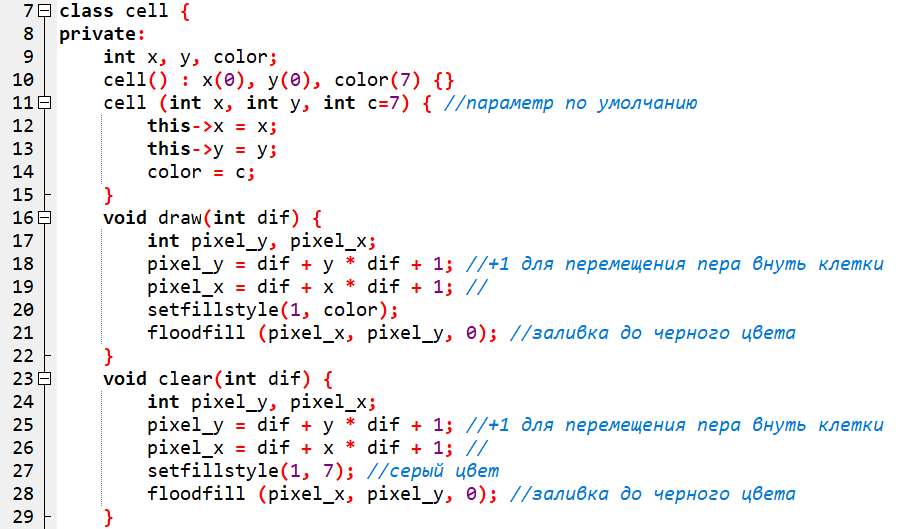
**Введение**

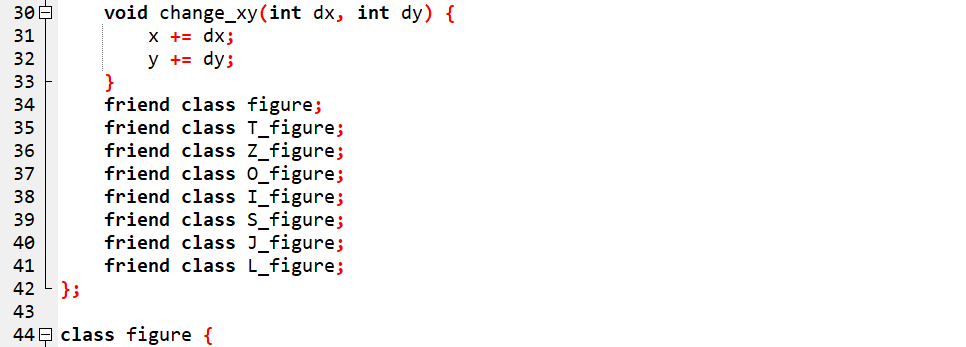
Необходимый минимум содержания работы:

1. Инкапсуляция (все поля данных не доступны из внешних функций).
2. Наследование (минимум 3 класса, один из которых - абстрактный).
3. Полиморфизм.
4. Конструкторы, Перегрузка конструкторов.
5. Списки инициализации.

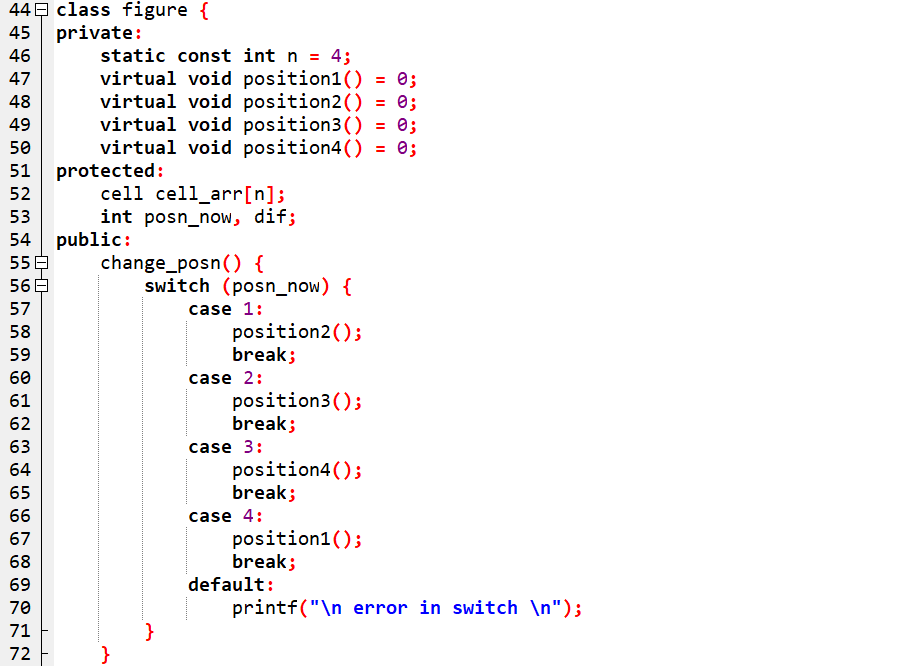
Я написал игру Тетрис с использованием объектно-ориентированных технологий: создан класс cell, который представляет собой клетку игрового поля, создан абстрактный класс figure, который служит родителем для классов фигур и использует класс cell. Проект программы состоит из четырех файлов: главного main.cpp, functions.h и functions.cpp для функций, classes.h для классов. Использована графическая библиотека graphics.h.  
  
Некоторые понятия, которые я использовал в этом отчете: игровое поле - поле для игры в Тетрис, 20 клеток в высоту, 10 в ширину; клетки поля - это то, из чего поле состоит, клетки изначально имеют серый цвет, разделены они черными линиями; фигуры - классические фигуры из тетрамино, которые используются в Тетрисе и имеют отличные от серого и черного цвета.

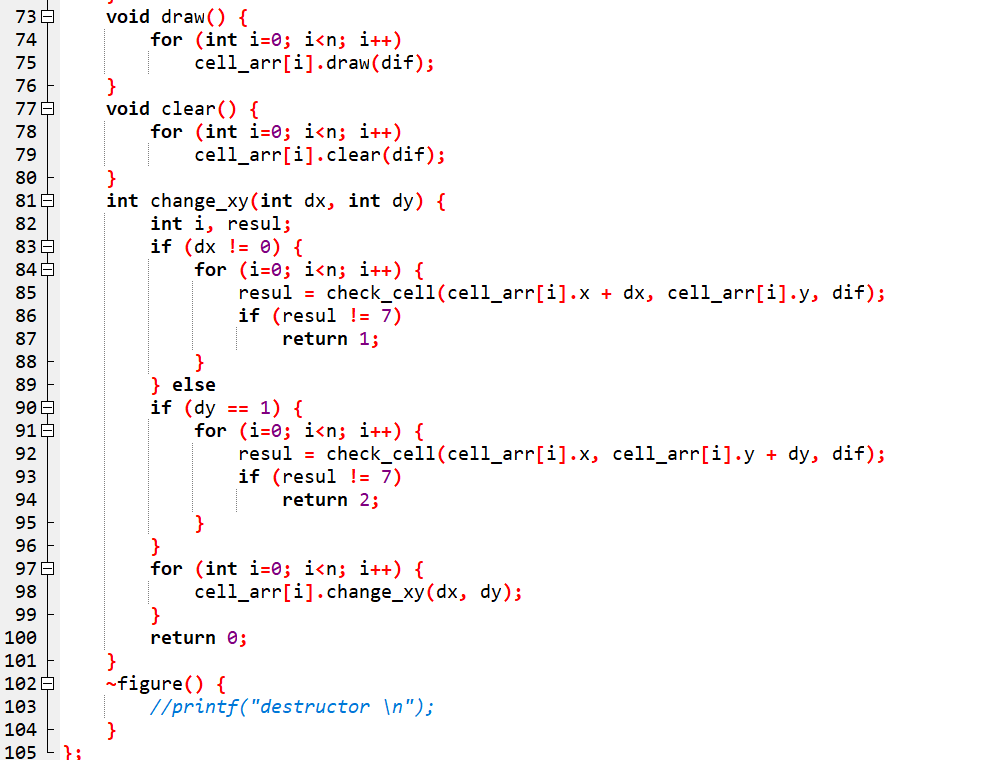
# **Классы**



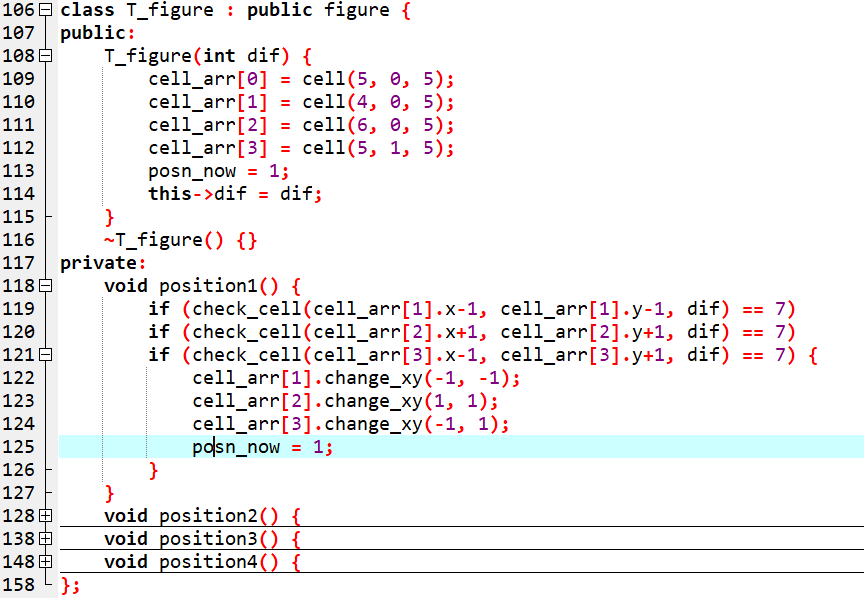


Как уже говорилось выше, класс cell представляет собой клетку игрового поля, поэтому он содержит координаты x и y, а также цвет (color) клетки. Есть перегрузка конструкторов. Первый конструктор содержит списки инициализации, второй конструктор содержит параметр по умолчанию. Метод draw() закрашивает клетку по координатам x, y на игровом поле в цвет из color. Метод clear() закрашивает клетку в изначальный серый цвет. Change\_xy() меняет координаты клетки. Все поля и методы класса приватны, однако класс дружественен всем классам фигур, что позволяет полноценно использовать его в этих классах.



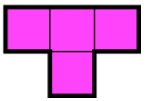


Класс figure является потомком для классов конкретных фигур. Так как каждая фигура в Тетрисе состоит из четырех клеток, я создал статическую переменную под это число. Виртуальные position1, position2 и другие меняют позицию фигуры (ее можно вращать на 90 градусов). Для каждой фигуры эти позиции реализуются по-разному, потому что каждая фигура уникальна. cell\_arr[] - это массив объектов cell, из которых фигура состоит. posn\_now - переменная для запоминания позиции. dif - длина и ширина клетки в пикселях. Методы draw(), clear() базируются на одноименных методах класса cell. Change\_xy() выполняет ту же функцию, однако он так же проверяет (с помощью функции check\_cell(), про которую будет рассказано дальше) правильность действий: нельзя перемещать фигуры за пределы игрового поля (возвращается значение 1). Также, если фигура достигла «дна» поля, то фигура перестает «падать» (возвращается 2).

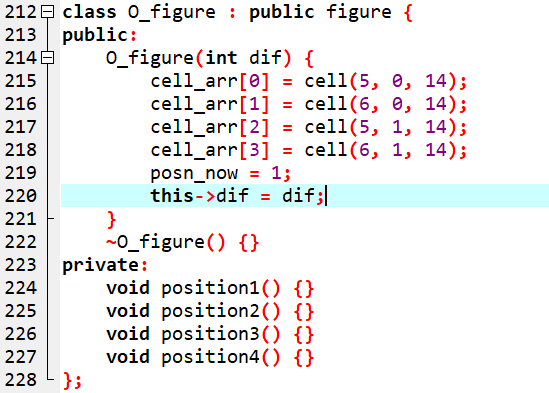


Т\_figure - потомок figure. Он используется для данной фигуры,

которая напоминает букву Т.

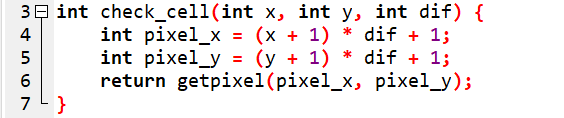


Конструктор задает координаты используемых клеток и устанавливает начальную позицию. Далее реализуются позиции: сначала идет проверка на возможность поворота фигуры и только после этого меняются координаты клеток и posn\_now. Методы position2, position3, position4 реализованы таким же способом. Другие классы фигур (Z, O, I, S, J, L) реализованы схожим образом с T-figure, несмотря на то, что у некоторых фигур существует только две или одна позиция.

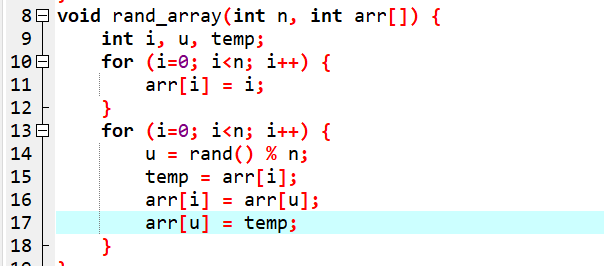


Других классов нет.

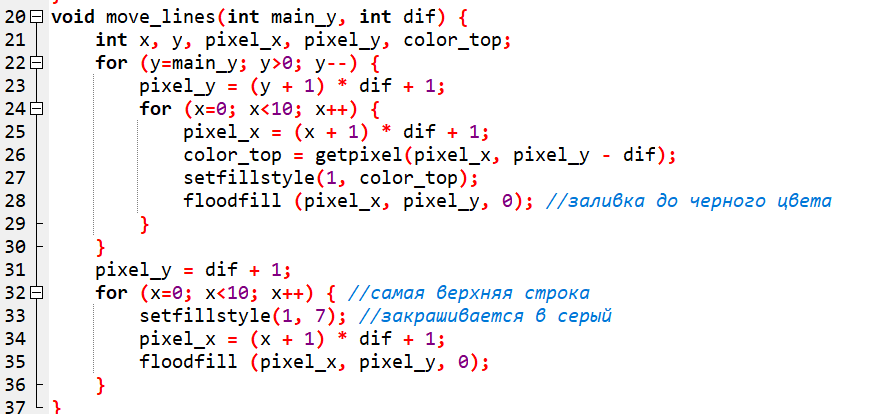
# **Функции**



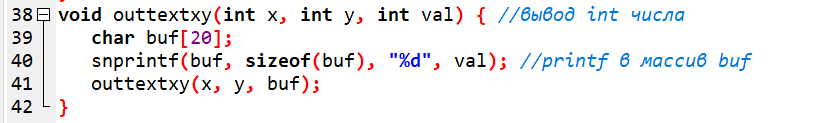
Данная функция принимает координаты и размер клетки, вычисляет координаты клетки в пикселях и возвращает цвет данной клетки. Эта функция используется в поворотах и передвижениях фигур, а также в другой функции.



Данная функция принимает массив (который используется в main.cpp) для рандомизации появления фигур. В ней индексы (1, 2, 3...) фигур (T, Z, O…) перемешиваются между собой с помощью rand(). Можно определить появление фигур исключительно с помощью rand(), однако в таком случае часто бывает ситуация, в которой одна-две фигуры не появляются много раз подряд.

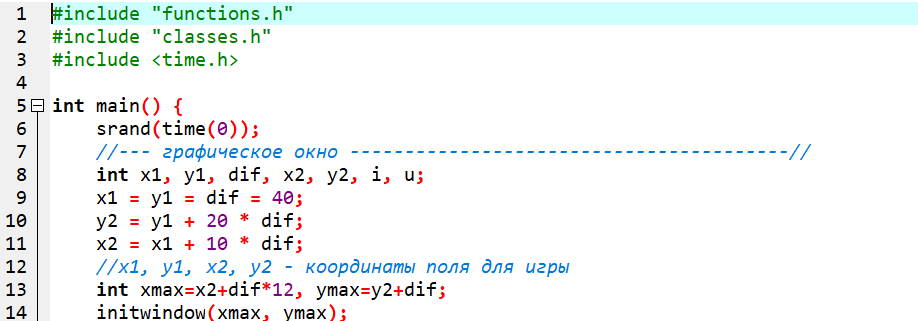


Данная функция используется, когда одна или несколько линий заполнена фигурами - по правилам игра эта (эти) линии удаляются и все верхние линии смещаются вниз. Здесь запоминаются и используются цвета, для которой нужна функция check\_cell().

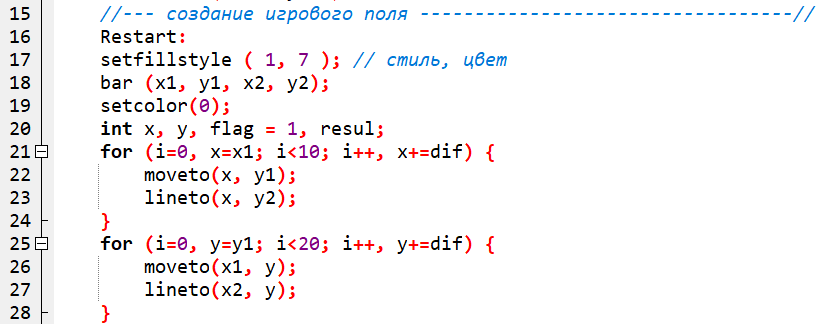


Данная функция является перегруженной по отношению к outtextxy(int, int, \*char) в библиотеке graphcs.h. С помощью функции snprintf() данные любого типа записываются в массив типа char. Этот массив используется в оригинальной outtextxt().

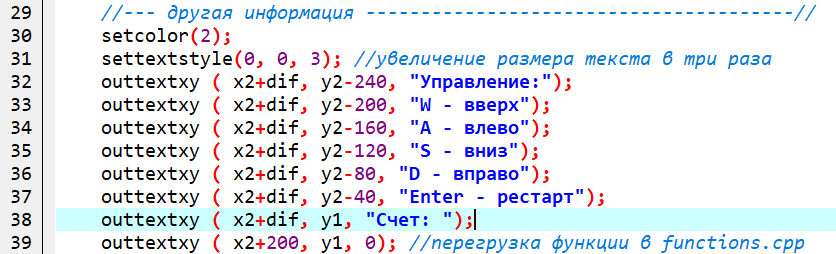
# **Алгоритм основной программы**



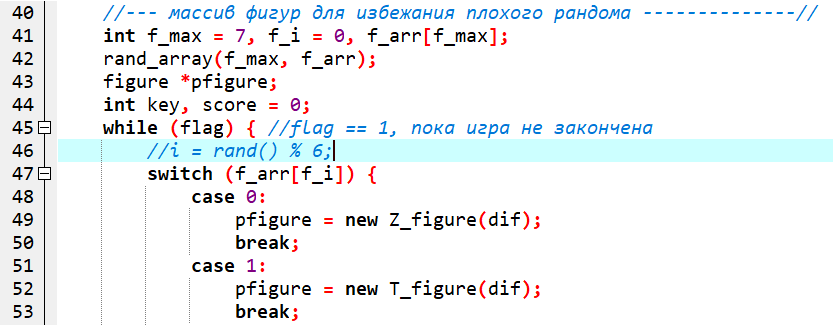
Размер клетки равен 40 пикселям и записывается в переменную dif. x1, y1, x2, y2 - координаты поля для игры. Размеры поля: 20 на 10, поэтому x2, y2 вычисляются таким образом. Графическое окно смещено вправо на несколько сотен пикселей для отображения другой информации.



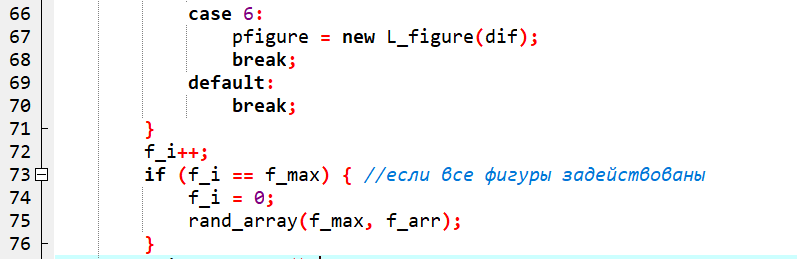
Restart - оператор для рестарта игры. Создается прямоугольник серого цвета, который потом за два цикла делится на клетки с помощью линий черного цвета.



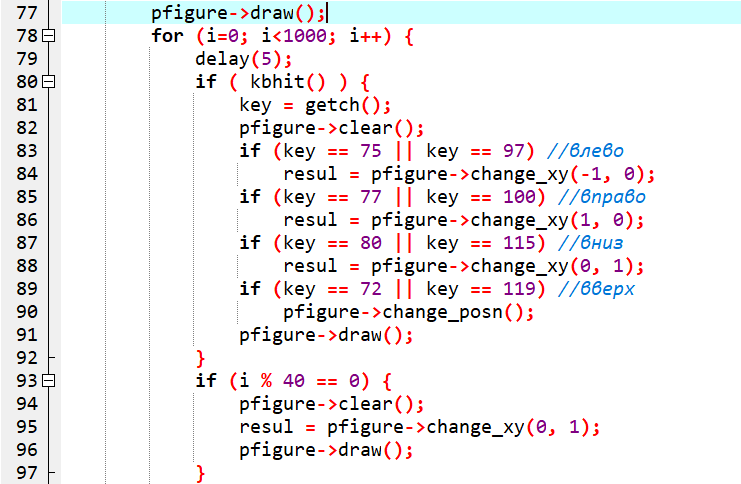
В данной части отображается информация об управлении и счете игрока - количество удаленных линий.



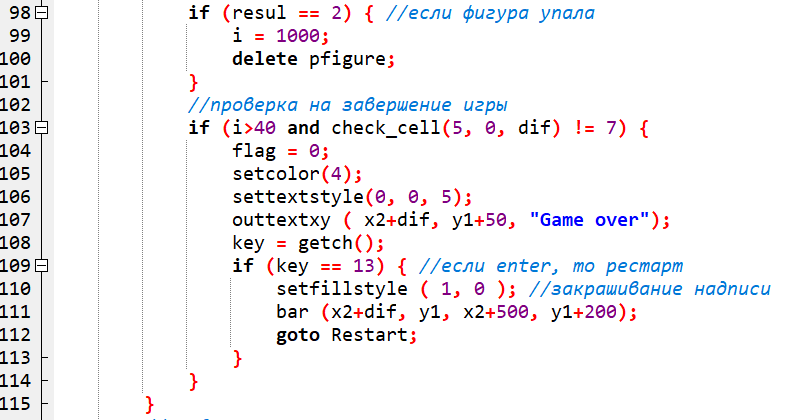
...



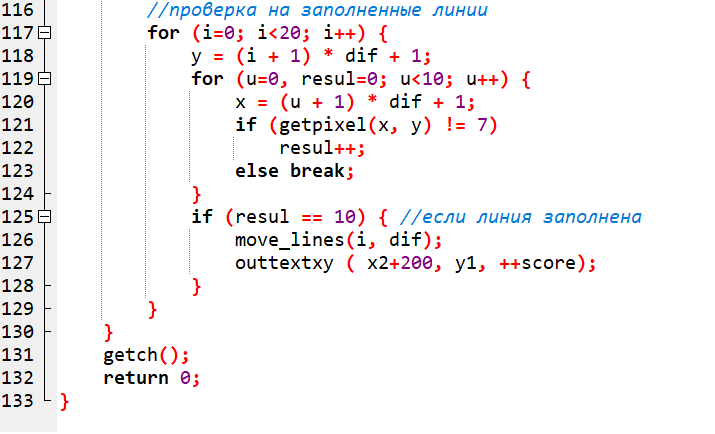
f\_arr[] - это массив для индексов фигур, в rand\_array() эти индексы перемешиваются. Далее запускается цикл, который заканчивается после того, когда нельзя поместить новую фигуру на игровом поле. С помощью switch() выбирается нужная фигура по индексу, здесь использован полиморфизм. Если все индексы использованы, то массив перемешивается заново.



В цикле запускается другой цикл - он используется до тех пор, пока фигура не упадет вниз. С помощью методов draw() и clear() фигура рисуется или стирается. Если нажата (kbhit() ) клавиши W, A, S или D, то соответсвенно координаты фигуры либо смещаются, либо она поворачивается на 90 градусов. Каждая 40-ая итерация принудительно смещает фигуру на одну клетку вниз.



Как говорилось выше, если change\_xy() возвращает 2, то это значит, что фигура упала на «дно» поля. Далле идет проверка на завершение игры. Координаты (5, 0) соответсвуют самой верхней клетке, в которой появляется любая из фигур. Здесь же можно начать игру заново, нажав на Enter.



Далее идет проверка на заполненные линии. Берется цвет каждой клетки в линии, и если все 10 клеток не серого цвета, то запускается move\_lines(). На этом цикл с flag и main.cpp заканчиваются.

**Скриншоты**

