

Санкт-Петербургский государственный университет  
информационных технологий, механики и оптики

Кафедра “Компьютерные технологии”

В.С. Джанмухамедов, А.П. Хвастунов, А.А. Шалыто

## **Визуализация сборки *Кубика Рубика***

Объектно-ориентированное программирование  
с явным выделением состояний

Проектная документация

Проект создан в рамках  
«Движения за открытую проектную документацию»  
<http://is.ifmo.ru>

Санкт-Петербург  
2005

## ОГЛАВЛЕНИЕ

Введение .....	3
1. Алгоритм сборки <i>Кубика Рубика</i> .....	4
2. Постановка задачи .....	14
3. Проектирование .....	15
3.1. Интерфейс пользователя .....	15
3.2. Диаграммы классов .....	18
3.3. Автоматы .....	19
3.3.1. Словесное описание автоматов .....	19
3.3.2. Схема связей и графы переходов автоматов .....	20
Заключение .....	31
Литература .....	32
Приложение 1. Протоколы работы программы .....	33
Приложение 2. Исходные коды .....	34

## Введение

Для алгоритмизации и программирования задач логического управления была предложена SWITCH-технология, названная также «автоматное программирование» или «программирование с явным выделением состояний» [1]. В дальнейшем этот подход был развит для объектно-ориентированных систем и назван «объектно-ориентированное программирование с явным выделением состояний». Подробно с указанным подходом и с примерами его использования можно ознакомиться на сайте <http://is.ifmo.ru>.

SWITCH-технология удобна для задач управления, когда требуется обеспечить правильность и надежность программы или процесса управления некоторым техническим объектом. Достоинствами рассматриваемой технологии являются централизация и прозрачность логики управления. Этому способствует также открытая проектная документация (<http://is.ifmo.ru>, раздел «Проекты»).

В данном проекте рассматриваемый подход применяется в другой области – для создания открытого проекта, посвященного автоматизации “человеческой” версии сборки *Кубика Рубика* и ее визуализации.

Известно, что из любого положения *Кубик Рубик* можно собрать не более чем за 22 поворота граней [2, 3]. Данную последовательность операций, возможно, найти лишь полным перебором. В нашем проекте представлен “человеческий” алгоритм сборки – последовательность четких указаний, при которых кубик будет собран из любого положения. В программу введен счетчик числа поворотов, который показывает, что используемый алгоритм приводит в среднем к 120 – 200 поворотам в зависимости от первоначального перемешивания.

Рассматриваемый пример хорош тем, что наглядно представляет пользователю такие понятия теории автоматов как состояния, переходы, входные условия и выходные воздействия. Поэтому этот пример хорошо подходит для первоначального знакомства с автоматным стилем программирования [4].

В примере для функционирования программы построено семь автоматов, пошагово собирающих кубик. Графы переходов всех автоматов планарные, что способствует их наглядности.

Для реализации проекта выбран язык *Visual C++*. Использовалась библиотека *MFC* для создания интерфейса программы и библиотека *OpenGL* для отображения сборки кубика на экране. Проект тестировался на операционной системе *Windows XP*.

## 1. Алгоритм сборки *Кубика Рубика*

Пронумеруем цвета граней кубика цифрами от единицы до шести. Будем считать, что первый цвет – цвет нижней грани, второй – передней, третий – правой, четвертой – задней, пятой – левой и, наконец, шестой - верхней (рис. 1). Каждой грани соответствует номер ее цвета.

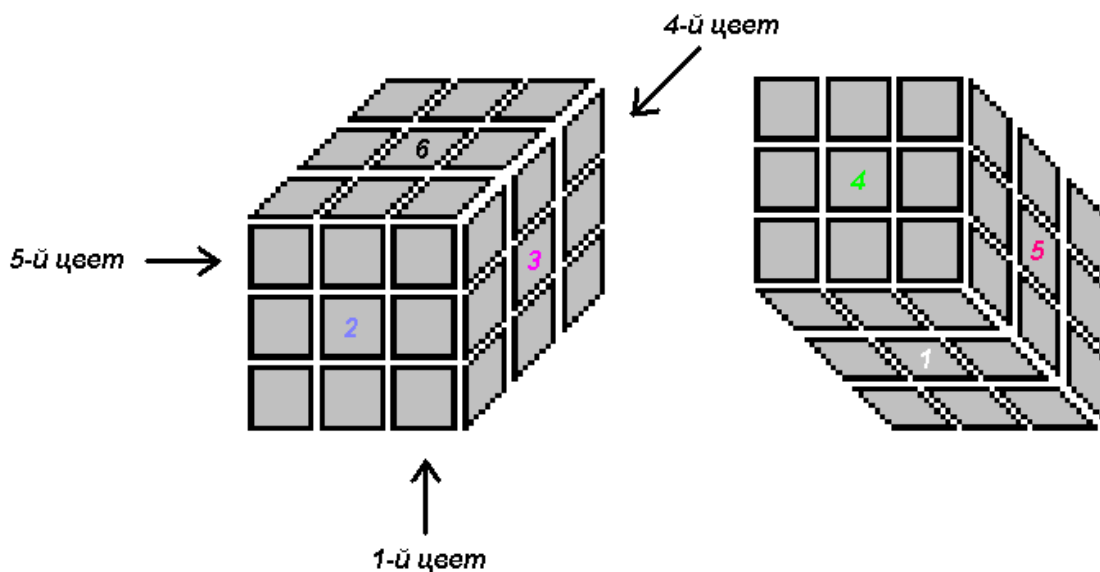


Рис. 1

Введем обозначения поворотов:

- $n$  – поворот на  $90^\circ$  по часовой стрелке грани с номером  $n$ ;
- $n'$  – поворот на  $90^\circ$  против часовой стрелки грани с номером  $n$ ;
- $n''$  – поворот на  $180^\circ$  грани с номером  $n$ .

Опишем шаги алгоритма.

### Шаг 1.

Выберем цвет нижней грани и зафиксируем его. Это будет первый цвет. Соберем крест внизу и по две одноцветные клетки на каждой боковой грани (рис. 2).

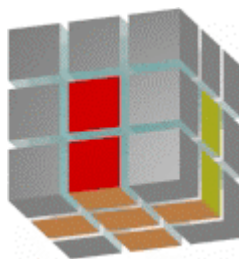
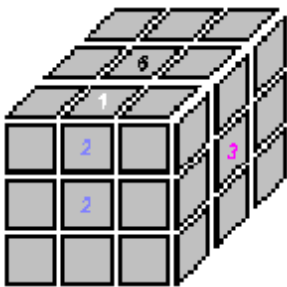

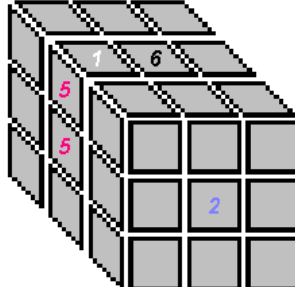
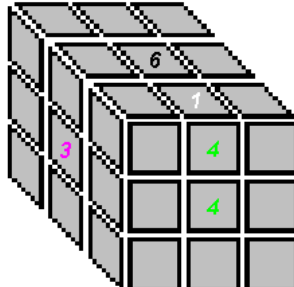
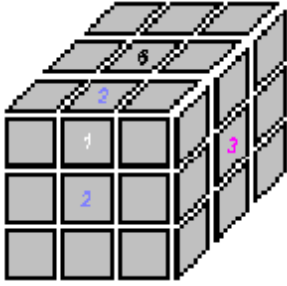
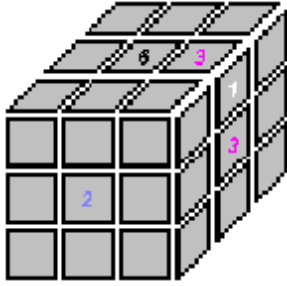
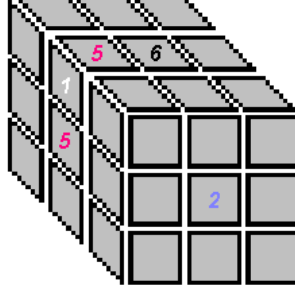
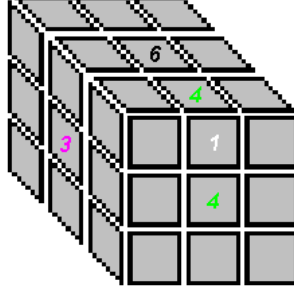
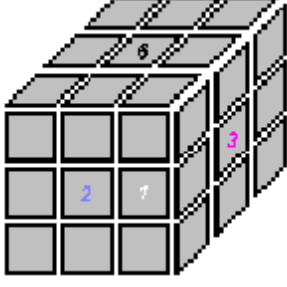
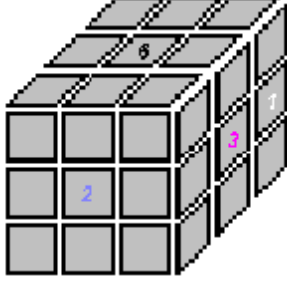
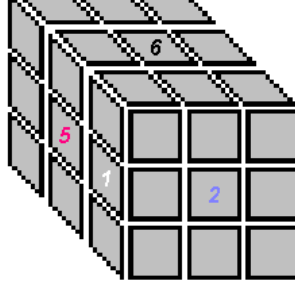
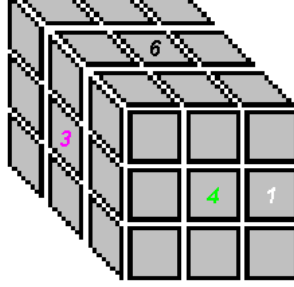
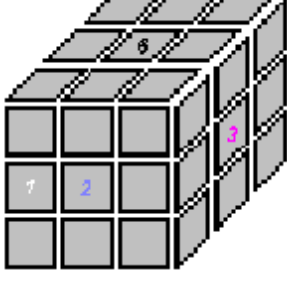
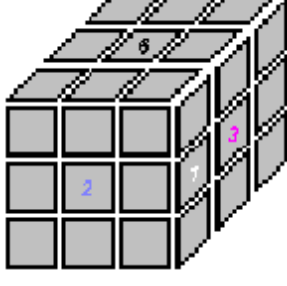
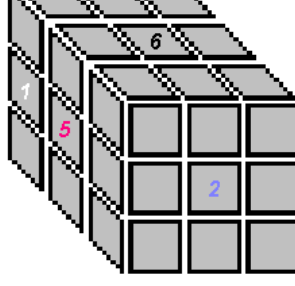
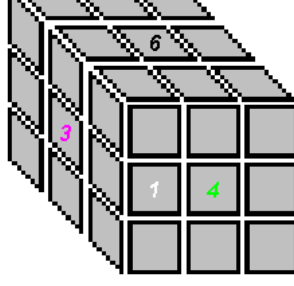
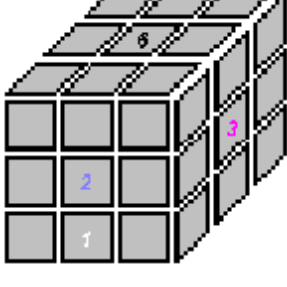
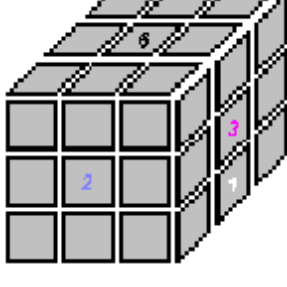
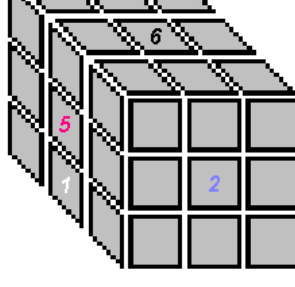
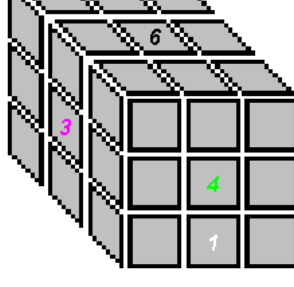


Рис. 2

Если кубик имеет расположение цветов, совпадающее с одной из конфигураций, приведенной на рис. 3, то выполняются действия, указанные в нижней части соответствующей клетки. Если ни одна из 24 конфигураций не подходит, то выполняется поворот верхней грани (грань номер 6) на  $90^\circ$  по часовой стрелке и описанная процедура повторяется сначала. Если произошло подряд четыре поворота верхней грани, то переходим на Шаг 2.

			
2''	3''	5''	4''
			
6'3'23	6'4'34	6'2'52	6'5'45
			
2'6'2	3'6'3	5'6'5	4'6'4
			
26'2'	36'3'	56'5'	46'4'
			
2''6'2''	3''6'3''	5''6'5''	4''6'4''

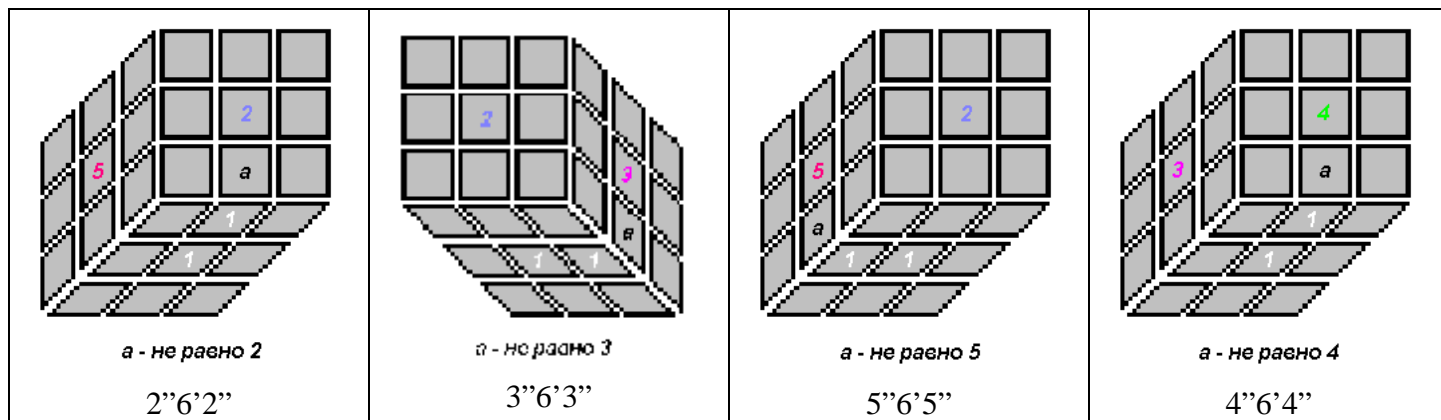


Рис. 3

## Шаг 2.

На этом шаге угловые клетки нижней грани размещаются на ней так, чтобы их цвета соответствовали цветам граней, к которым они примыкают (рис. 4).

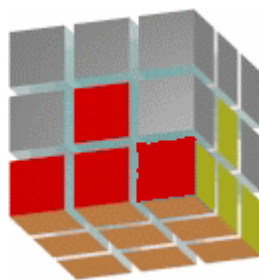
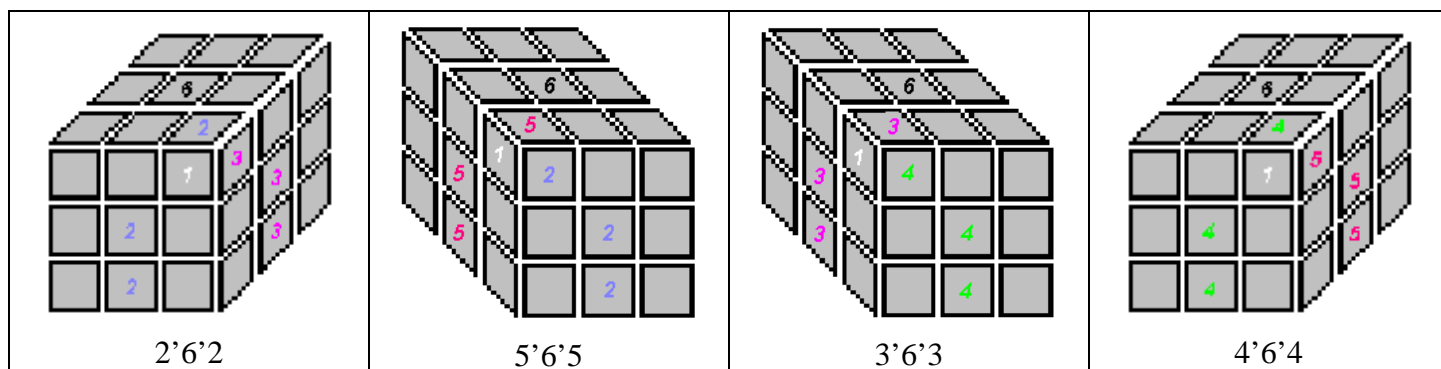
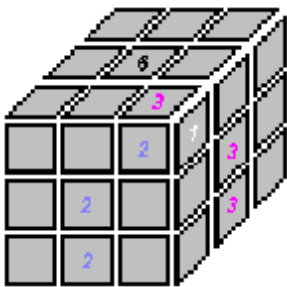
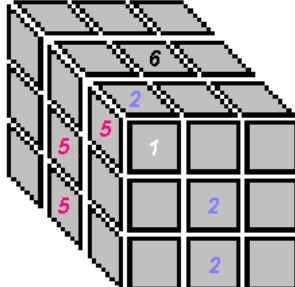
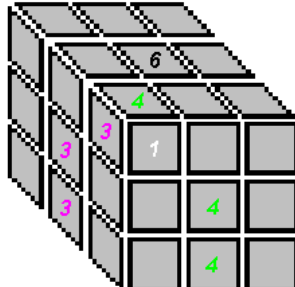
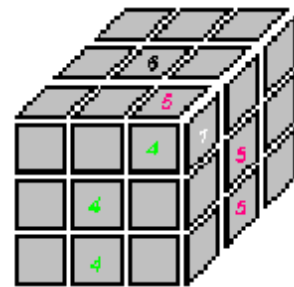
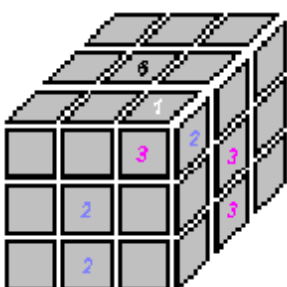
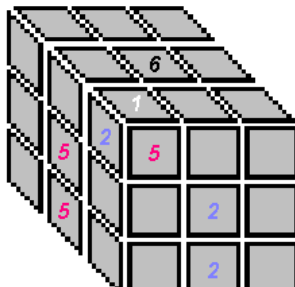
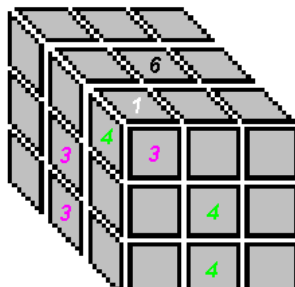
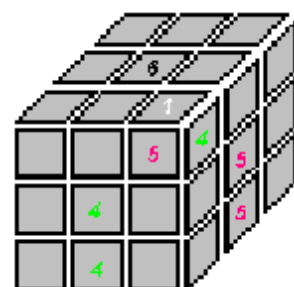
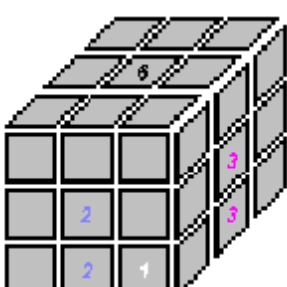
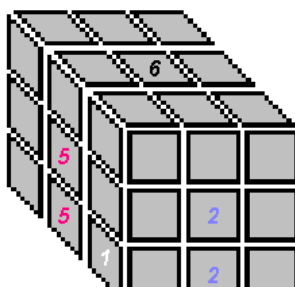
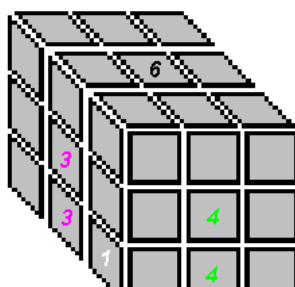
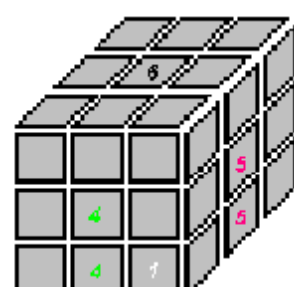
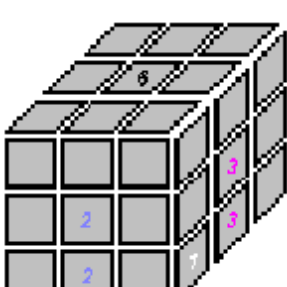
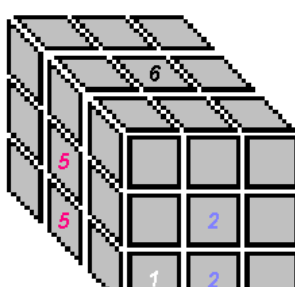
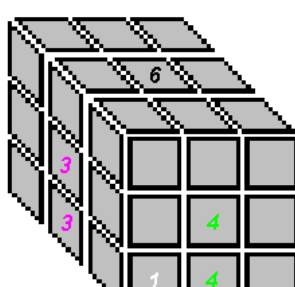
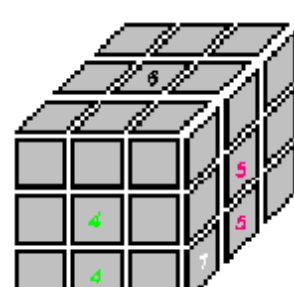


Рис. 4

Как и на Шаге 1, если кубик принимает одну из конфигураций (рис. 5), выполняются соответствующие действия. Если ни одна из 24 конфигураций не подходит, то выполняется поворот верхней грани (грань номер шесть) на  $90^\circ$  по часовой стрелке и описанная процедура повторяется сначала. Если произошло подряд четыре поворота верхней грани, то переходим на Шаг 3.



 <p>363'</p>	 <p>262'</p>	 <p>464'</p>	 <p>565'</p>
 <p>36'3'6''363'</p>	 <p>26'2'6''262'</p>	 <p>46'4'6''464'</p>	 <p>56'5'6''565'</p>
 <p>2'6'2</p>	 <p>5'6'5</p>	 <p>3'6'3</p>	 <p>4'6'4</p>
 <p>2'6'2</p>	 <p>5'6'5</p>	 <p>3'6'3</p>	 <p>4'6'4</p>

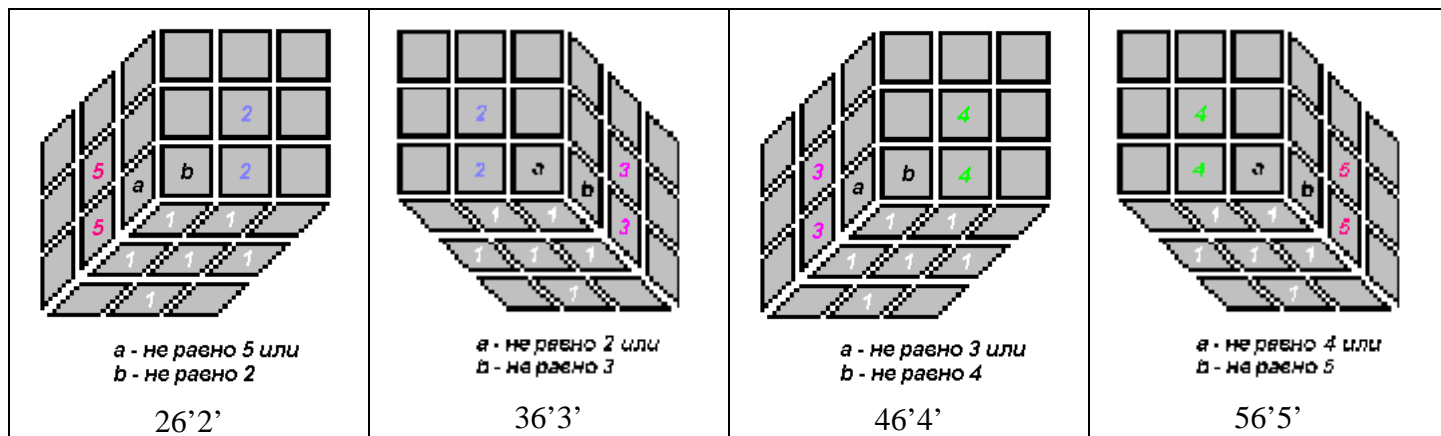


Рис. 5

### Шаг 3.

Соберем нижний и средний пояса кубика (рис. 6).

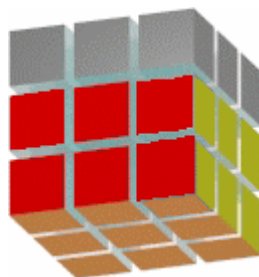
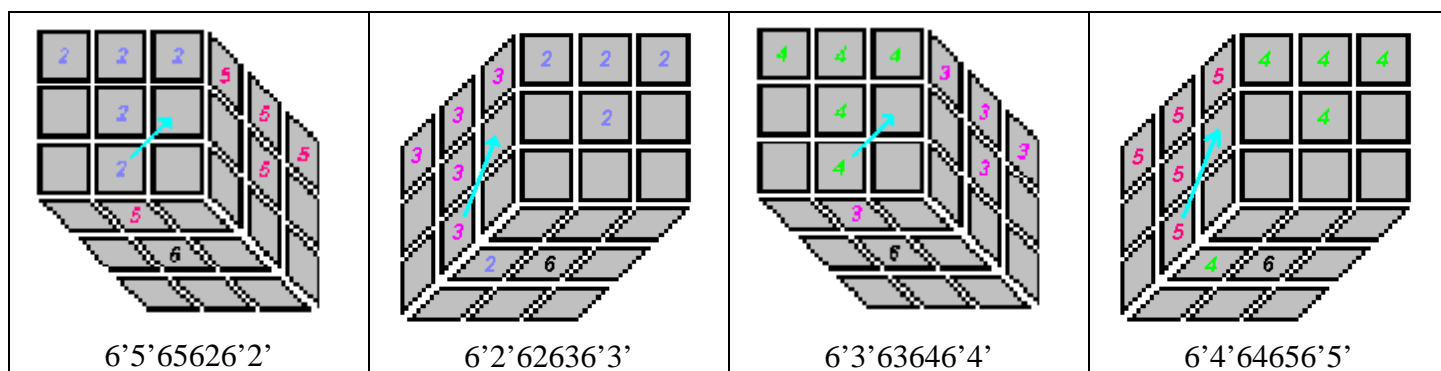


Рис. 6

Если кубик принимает одну из конфигураций (рис. 7), то выполняются соответствующие действия. Если ни одна из восьми конфигураций не подходит, то выполняется поворот верхней грани (грань номер шесть) на 90° по часовой стрелке и описанная процедура повторяется сначала.

На этом и следующем рисунках каждая стрелка показывает место, куда встанет маленький кубик (ячейка кубика в целом) после поворотов.





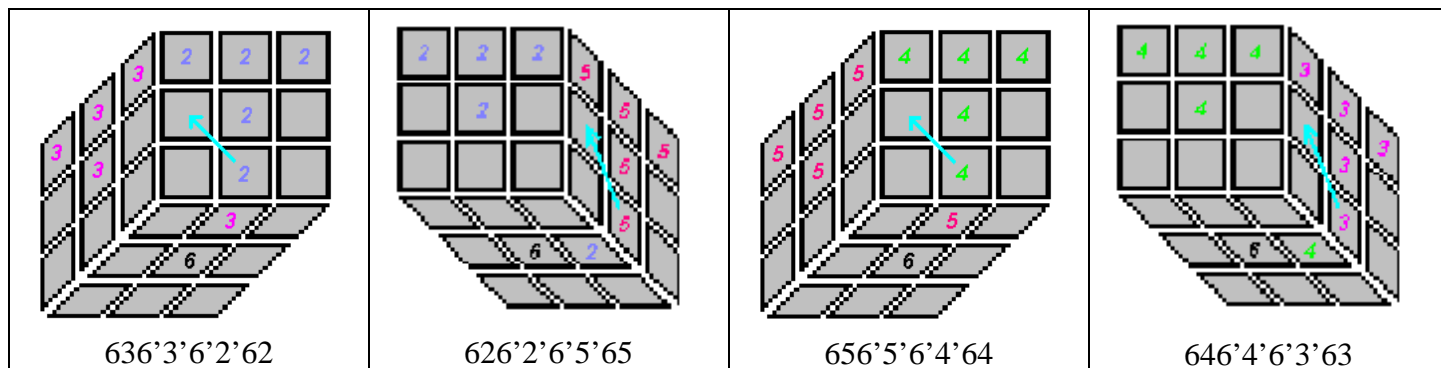


Рис. 7

Если произошло подряд четыре поворота верхней грани, и при этом кубик принимает одну из конфигурацию, из представленных на рис. 8, то выполняются соответствующие действия и процедура начинается опять с самого начала. В противном случае, переходим к Шагу 4.

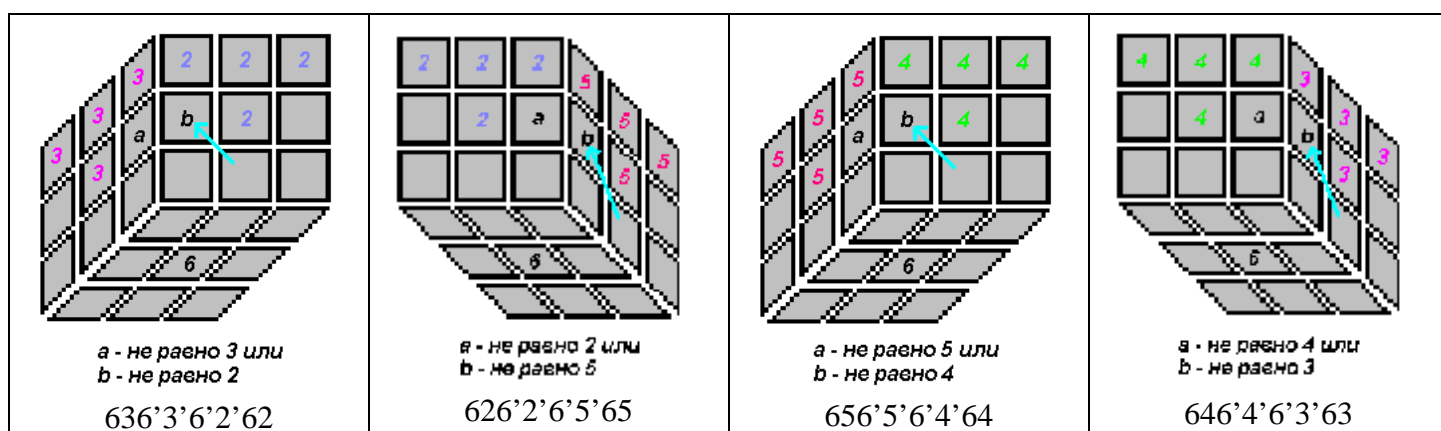


Рис. 8

#### Шаг 4.

Ориентируем боковые (не угловые!) маленькие кубики верхней грани так, чтобы на ней образовался крест (рис. 9).

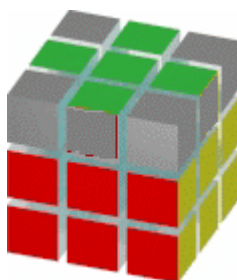


Рис. 9

Если кубик принимает одну из конфигураций (рис. 10), то выполняются соответствующие действия. Потом переходим к Шагу 5.

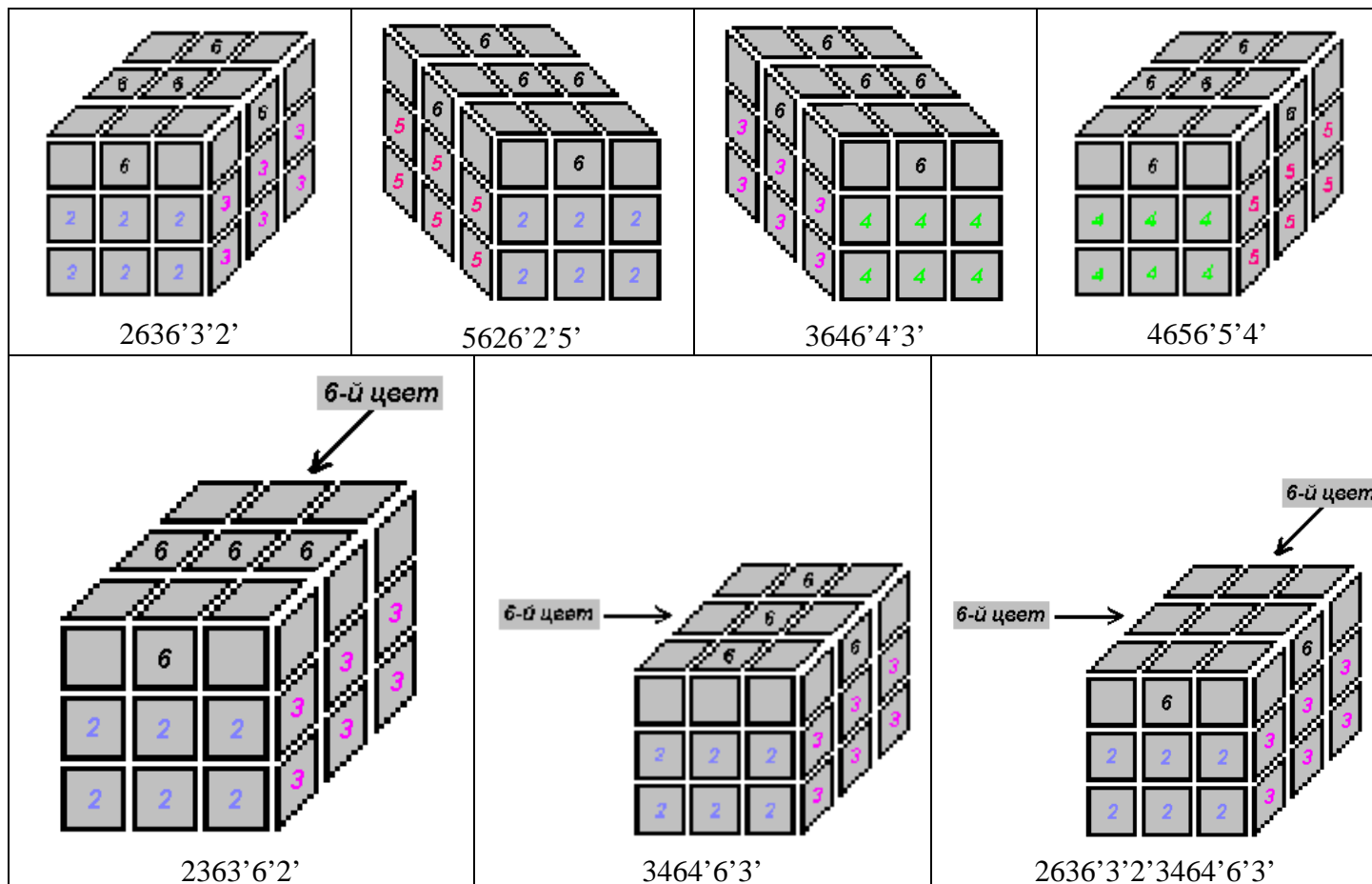


Рис. 10

Отметим, что первоначально в алгоритме была ошибка, связанная с тем, что в известных источниках, например [5-7], последний из приведенных случаев не рассматривался, а было сказано, что он аналогичен одному из предшествующих.

### Шаг 5.

Поставим боковые (не угловые!) маленькие кубики верхней грани так, чтобы цвета на них соответствовали граням, к которым они примыкают (рис. 11).

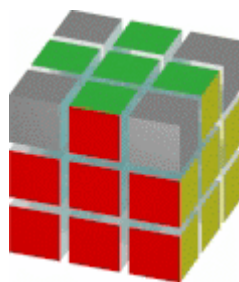


Рис. 11

Если кубик принимает одну из конфигурацию (рис. 12), то выполняются соответствующие действия.

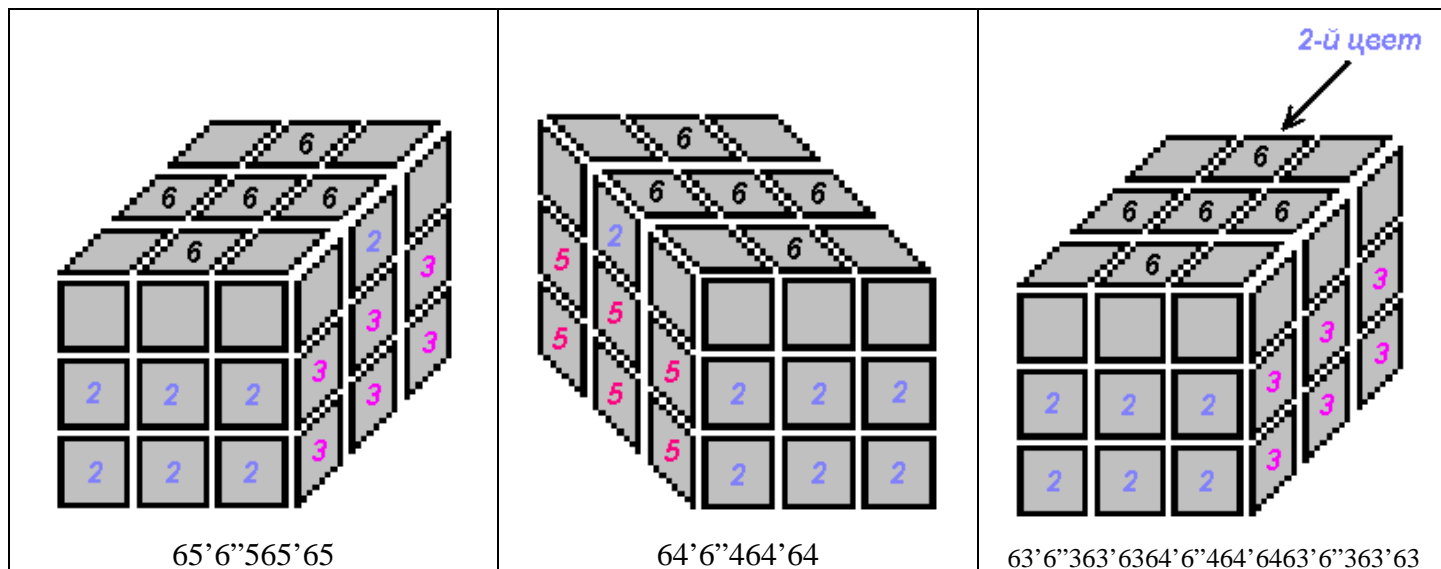


Рис. 12

Затем ищутся конфигурации, которые принимает кубик (рис. 13), и выполняются соответствующие действия.

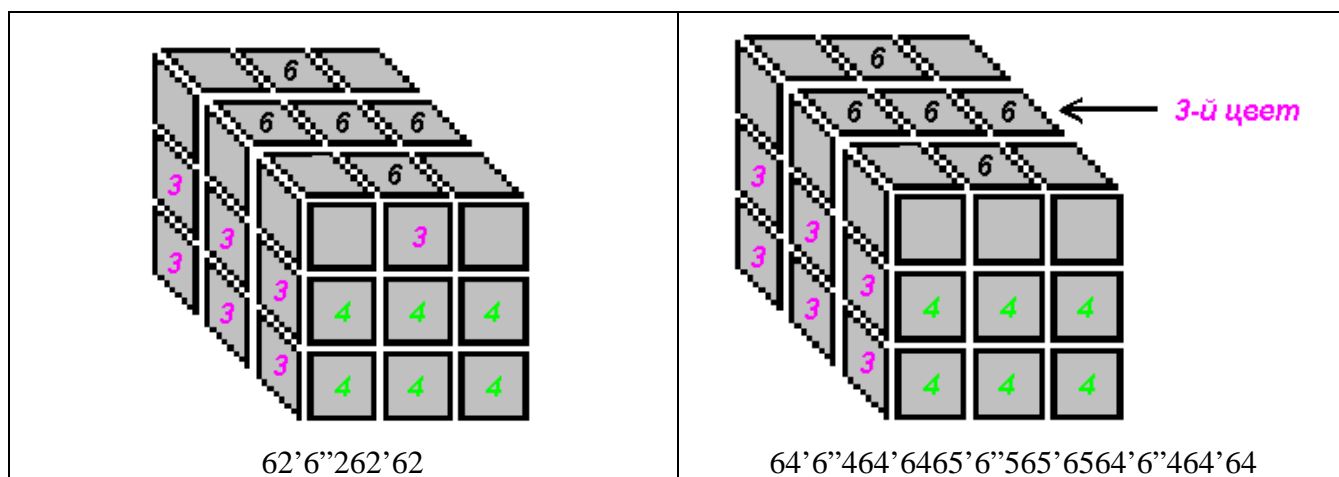


Рис. 13

Наконец, такие же действия производятся для конфигурации на рис. 14. В конце выполняется переход к Шагу 6.

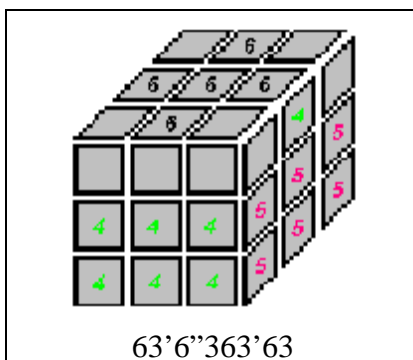


Рис. 14

### Шаг 6.

Разместим угловые маленькие кубики верхней грани так, чтобы цвета на них соответствовали цветам граней, к которым они примыкают (рис. 15).



Рис. 15

Если кубик принимает одну из конфигураций (рис. 16), то выполняются соответствующие действия.

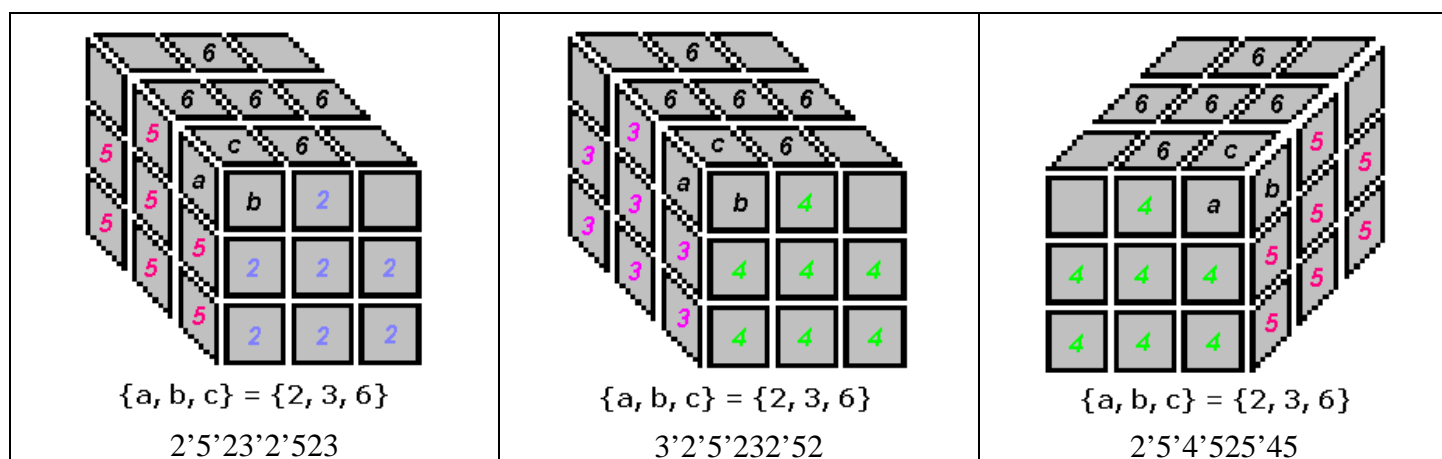


Рис. 16

В противном случае, ищутся конфигурации, которые принимает кубик (рис. 17), и выполняются необходимые действия для них. В конце производится переход к Шагу 7.

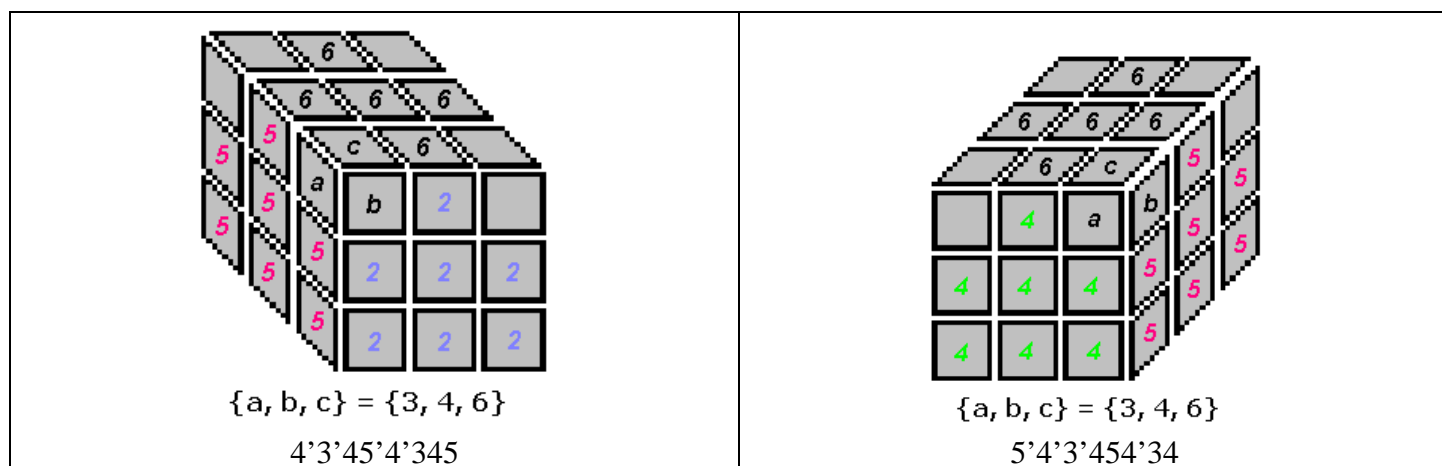


Рис. 17

### Шаг 7.

На этом шаге кубик собирается полностью (рис. 18).



Рис. 18

Проверим, принимает ли кубик одну из конфигураций на рис. 19. Если да, то выполняются соответствующие действия. В противном случае, поворачивается верхняя грань (грань номер шесть) на  $90^\circ$  по часовой стрелке. Снова выполняется проверка, принимает ли кубик одну из конфигураций на рис. 19. При положительном исходе выполняются необходимые действия, а потом поворачивается верхняя грань. Снова проверяется совпадение с рис. 19 и при положительном исходе выполняются необходимые действия, а потом поворачивается верхняя грань. В последний раз проверяется совпадение с рис. 19 и при положительном исходе выполняются необходимые действия.

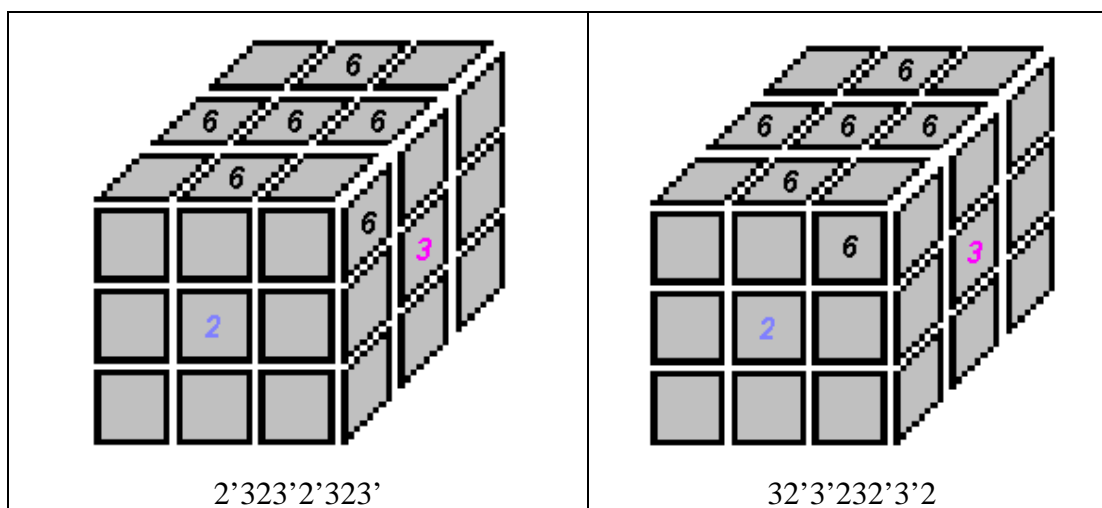


Рис. 19

Верхняя грань кубика поворачивается по часовой стрелке до тех пор, пока кубик не будет собран полностью.

## 2. Постановка задачи

Цель данной работы состоит в создании программы, которая наглядно показывает автоматическую сборку *Кубика Рубика*. При этом программа предоставляет пользователю следующие возможности:

- поворачивать кубик вокруг своего центра в трех плоскостях;
- случайно перемешивать кубик (сначала он в собранном состоянии);
- автоматически собирать кубик маленькими и большими шагами;
- сохранять позиции или загружать их из файла;
- сохранять все сделанные операции в файл протокола работы.

## 3. Проектирование

### 3.1. Интерфейс пользователя

Рассмотрим пользовательский интерфейс. Основным рабочим окном проекта является представленное ниже диалоговое окно (рис. 20).

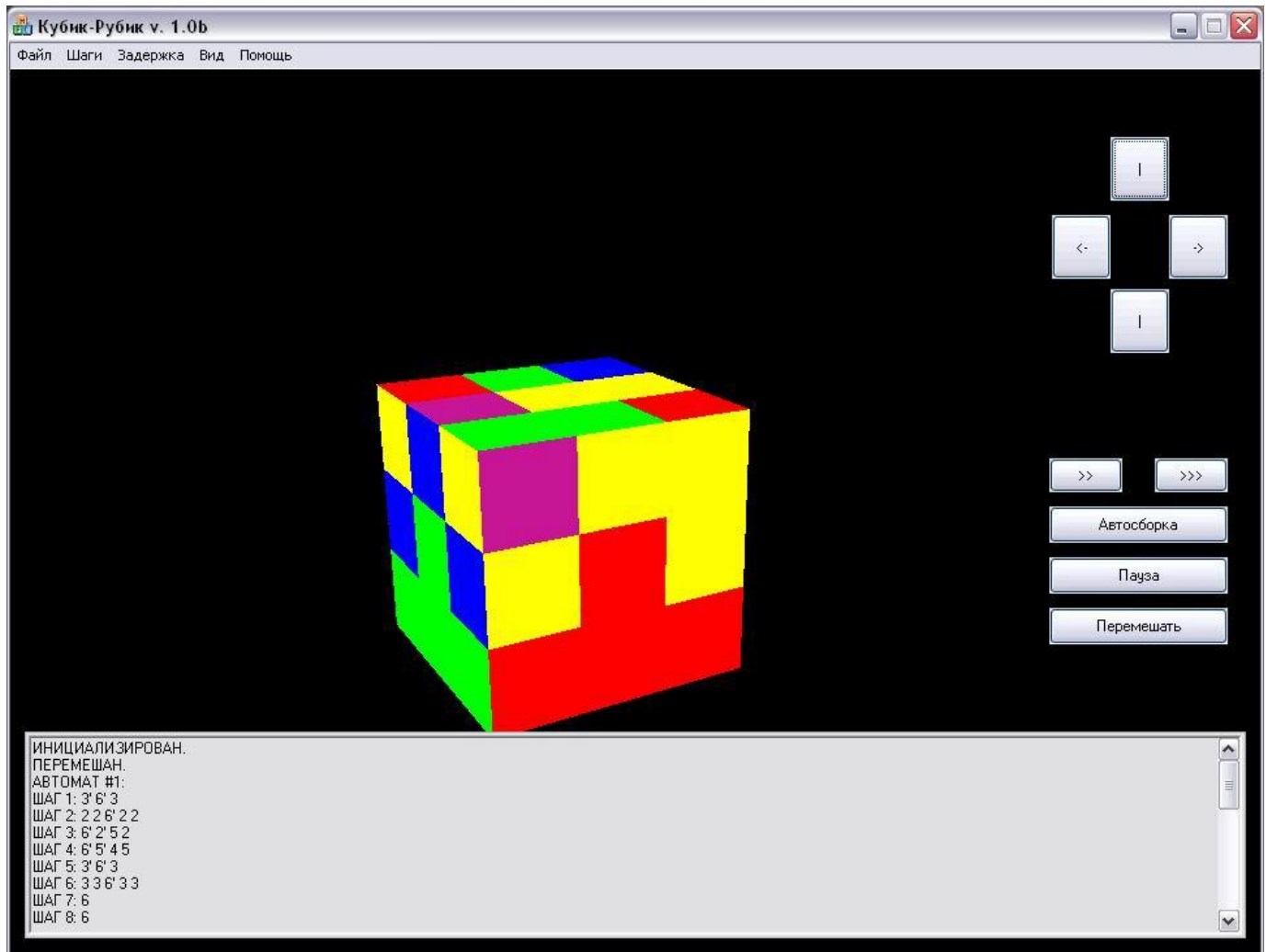


Рис. 20. Главное окно

В центральной части окна визуализируется модель кубика.

Справа в окне находятся две группы кнопок управления. Первая (располагается сверху) отвечает за вращения кубика по всем трем осям для более наглядного понимания алгоритма, а вторая (располагается снизу) – за выполнение алгоритма, реализованного системой из семи последовательно работающих автоматов.

Кнопки первой группы позволяют вращать кубик вверх и вниз, влево и вправо.

Опишем функциональность кнопок второй группы.


Кнопка «>>» обеспечивает переход из одного состояния в другое в пределах одного автомата, а кнопка «>>>» позволяет перейти между автоматами. При этом отметим, что при переходах в пределах одного автомата могут происходить более одного поворота граней кубика.

При нажатии кнопки *Автосборка* запускается алгоритм сборки кубика с задержкой между поворотами граней, выбранной в меню *Задержка*, описываемое ниже. При этом все семь автоматов «отработают» без остановки и кубик перейдет из произвольного исходного состояния в «собранный».

Действие кнопки *Пауза* соответствует ее названию – при ее нажатии останавливается сборка кубика, а надпись на кнопке изменяется на *Продолжить*. При повторном нажатии происходит обратное действие (сборка продолжается, надпись изменяется на «*Пауза*»).

Кнопка *Перемешать* предназначена для случайного перемешивания кубика путем произвольных поворотов граней в любом направлении.

В нижней части окна выводится протокол работы программы (рис. 21). Описание формата приведено в Приложении 1.



```
ИНИЦИАЛИЗИРОВАН.  
ПЕРЕМЕШАН.  
АВТОМАТ #1:  
ШАГ 1: 3' 6' 3  
ШАГ 2: 2 2 6' 2 2  
ШАГ 3: 6' 2' 5 2  
ШАГ 4: 6' 5' 4 5  
ШАГ 5: 3' 6' 3  
ШАГ 6: 3 3 6' 3 3  
ШАГ 7: 6  
ШАГ 8: 6
```

Рис. 21. Протокол

Главное окно программы также содержит меню, включающее следующие пункты:

- *Файл* (рис. 22) содержит следующие команды: *Новый* (сбрасывает автомат в начальное состояние и отображает кубик в собранном виде), *Открыть* (загружает из файла сохраненное ранее состояние кубика), *Сохранить* (сохраняет в файл текущее состояние кубика), *Выход* (выход из программы).

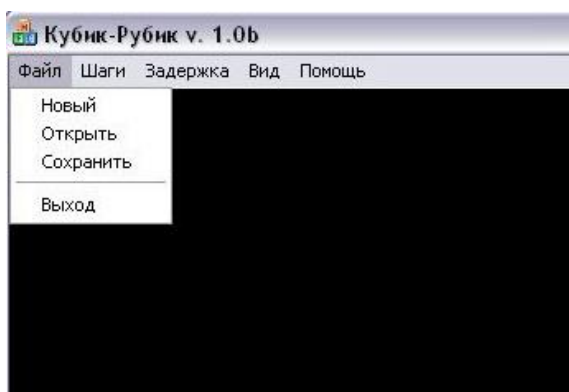


Рис. 22. Меню *Файл*

- *Шаги* (рис. 23) содержит команды *Маленький шаг* (выполняет переход между состояниями одного автомата – совершает действия аналогичные кнопке «>>>»), *Большой шаг* (выполняет переход между автоматами – выполняет действия аналогичные кнопке «>>>>») и *Автосборка* (автоматическая сборка кубика – аналогично как это выполняется при нажатии кнопки *Автосборка*).



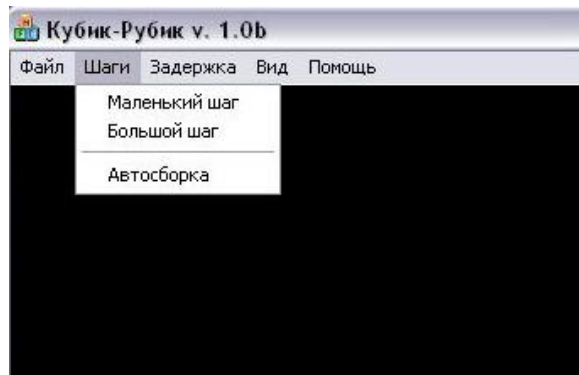


Рис. 23. Меню *Шаги*

- *Задержка* (рис. 24) – устанавливает одно из пяти (100 мс, 200 мс, 300 мс, 400 мс или 500 мс) возможных значений для величины задержки между поворотами граней.

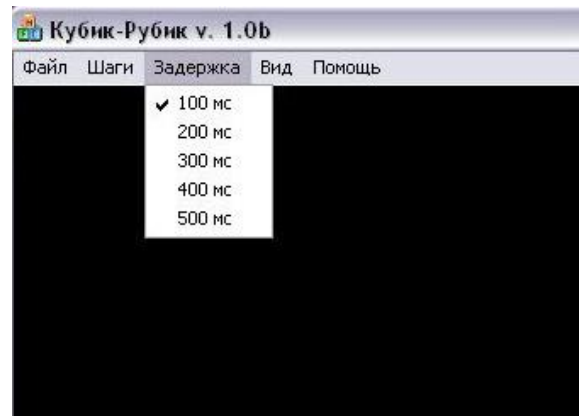


Рис. 24. Меню *Задержка*

- *Вид* – содержит единственный пункт, позволяющий показывать или скрывать протокол работы (лог).
- *Помощь* – в нем находится пункт *О программе*, содержащий информацию о программе и ее авторах.

На рис. 25 показан момент окончания сборки кубика, с указанием количества поворотов, осуществленных программой из выбранного случайного начального положения.

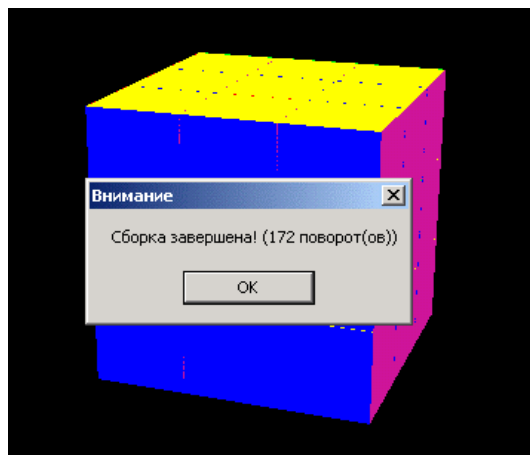


Рис. 25

## 3.2. Диаграмма классов

Главным классом проекта является класс `CBrick0_1App`. Этот класс отвечает за запуск, работу и начальную инициализацию проекта.

Класс `CBrick0_1Dlg` обеспечивает визуализацию программы. Этот класс является наследником стандартного класса `CDialog`. В нем описана обработка всех событий при работе с интерфейсом программы. Одним из полей данного класса является экземпляр класса `CBigBrick`, взаимодействие с которым обеспечивает отрисовку самого *Кубика Рубика* на экране.

Класс `CBigBrick` полностью описывает структуру *Кубика Рубика*. В его состав входят 27 элементов класса `CsmallBrick`. В итоге кубик разбивается на 27 маленьких кубиков. Также данный класс содержит методы, отвечающие за повороты граней, которые влекут за собой пере-строение структуры кубика.

`CsmallBrick` – класс для маленького кубика, входящего в состав большого (`CBigBrick`). Данный класс имеет поля, которые полностью описывают его координаты в пространстве и цвета всех шести граней. В состав класса входят методы поворотов, аналогичные методам из вышеуказанного класса.

Класс `CAutoBrick` реализует все автоматы (описание смотрите в разд. 3.3).

Кроме пяти основных классов программа содержит вспомогательный класс `CCol`, в котором реализованы цвета граней кубика.

Диаграмма классов представлена на рис. 26.

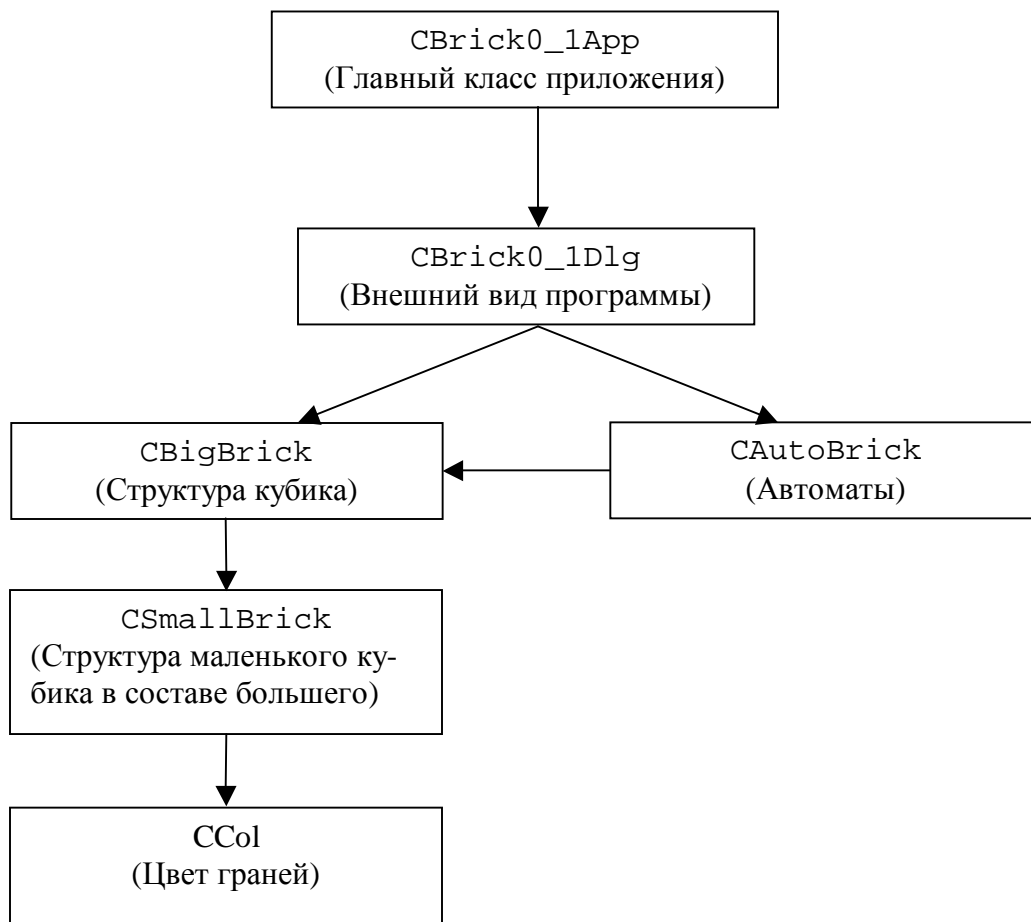


Рис. 26. Диаграмма классов

### 3.3. Автоматы

#### 3.3.1. Словесное описание автоматов

Алгоритм сборки кубика реализован семью последовательными автоматами. Полное описание алгоритма смотрите в разд. 1.

1. Автомат *AI\_1* собирает нижний крест с боковыми кубиками (рис. 27).

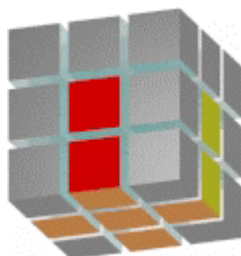


Рис. 27

2. Автомат *AI\_2* собирает полный нижний слой и часть среднего (рис. 28).

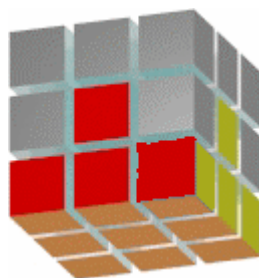


Рис. 28

3. Автомат *AI\_3* собирает целиком два нижних слоя (рис. 29).

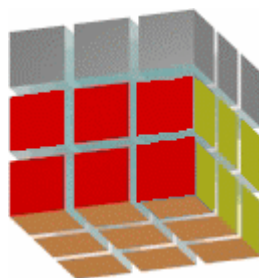


Рис. 29

- 4-5. Автоматы *AI\_4* и *AI\_5* собирают верхний крест (рис. 30).

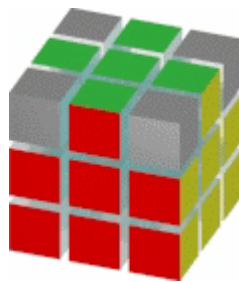


Рис. 30

6. Автомат  $A1\_6$  ставит угловые кубики верхней грани на свои места (рис. 31).



Рис. 31

7. Правильно ориентирует угловые кубики верхней грани, окончательно собирая кубик (рис. 32).



Рис. 32

### 3.3.2. Схемы связей и графы переходов автоматов

Ниже приведены схемы связей и графы переходов семи автоматов (рис. 33 - 46), в которых  $x1, \dots, x7$  – входные воздействия (условия переходов), а  $z1, z2, \dots, z25$  – выходные воздействия. Каждый автомат соответствует одному шагу алгоритма.

Переменные  $x1, \dots, x7$  являются массивами, содержащими номера тех конфигураций, которым удовлетворяет кубик на данном этапе. Все конфигурации подробно описаны в разд. 1. При этом в первых трех автоматах дополнительно введены конфигурации с номерами 25, 25 и 13 соответственно. Они соответствуют тому, что кубик не удовлетворяет ни одной другой конфигурации.

Под записью « $x1 = 2$ » будем понимать « $2 \in x1$ », то есть 2 входит в массив  $x1$ .

Условные обозначения поворотов:

1. Цифра обозначает номер грани.

2. Отсутствие штриха означает поворот грани по часовой стрелке на  $90^\circ$ , один штрих – поворот против часовой стрелки на  $90^\circ$ , двойной штрих – поворот на  $180^\circ$ .

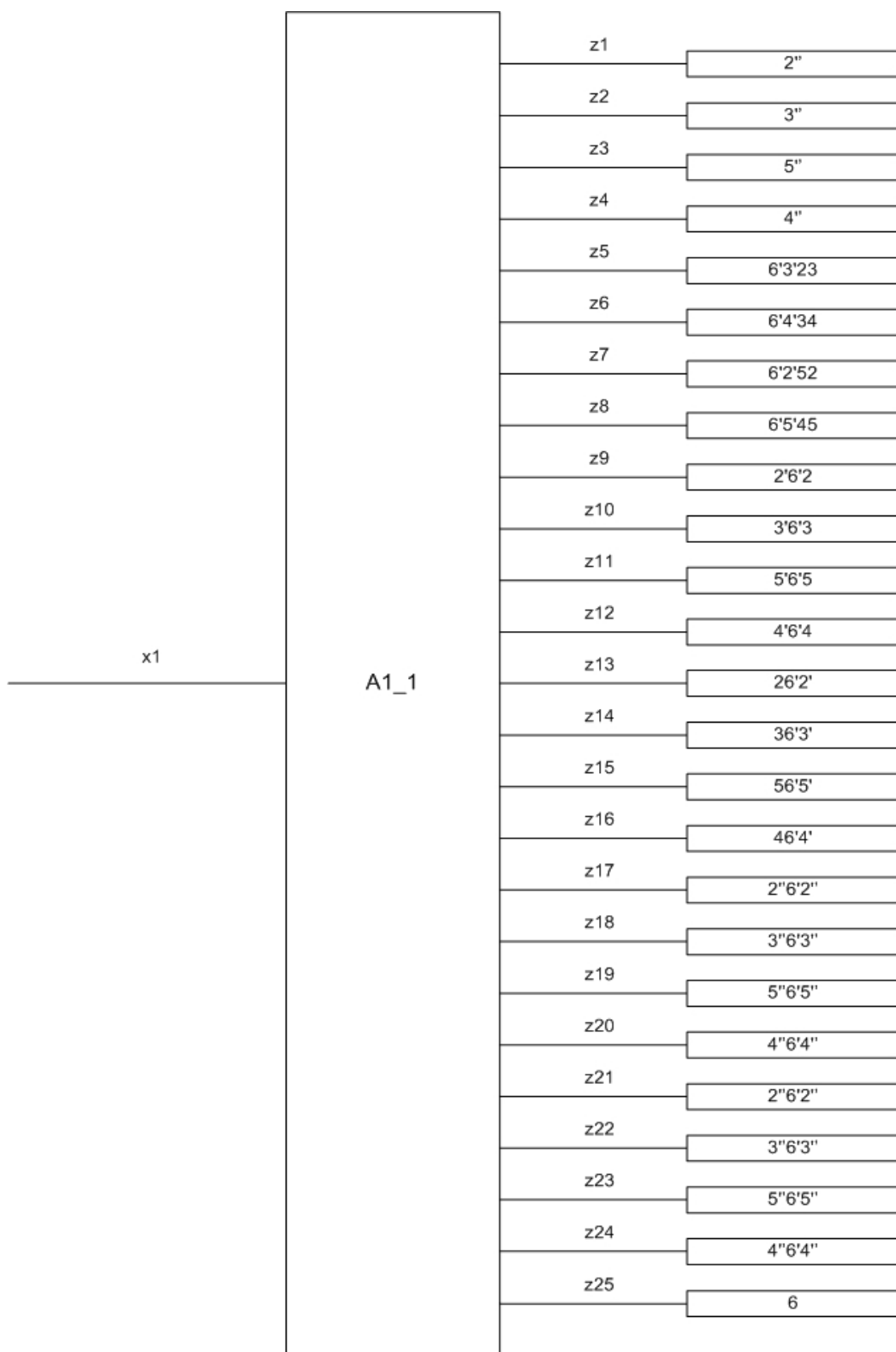


Рис. 33. Автомат 1

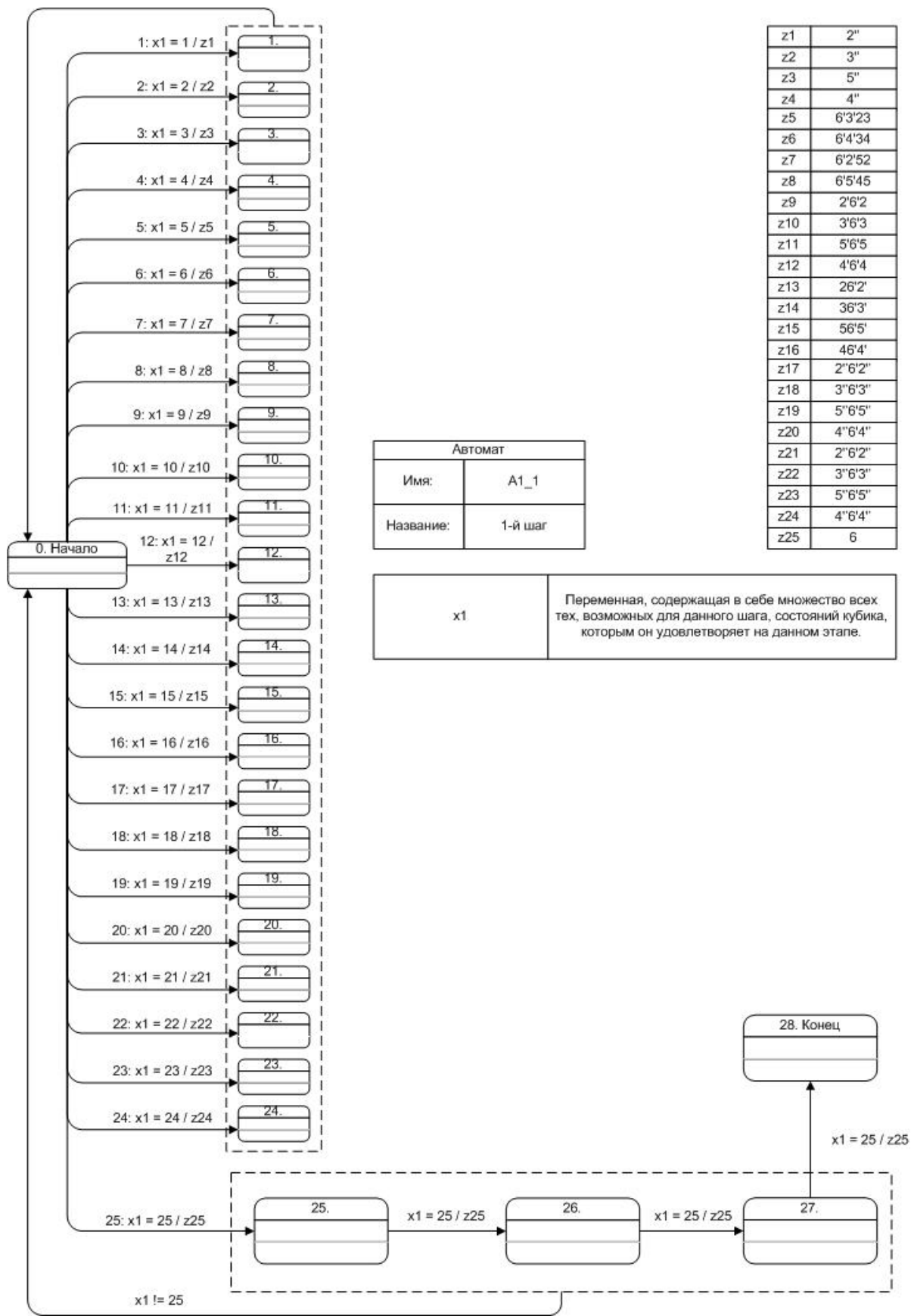


Рис. 34. Схема переходов автомата 1

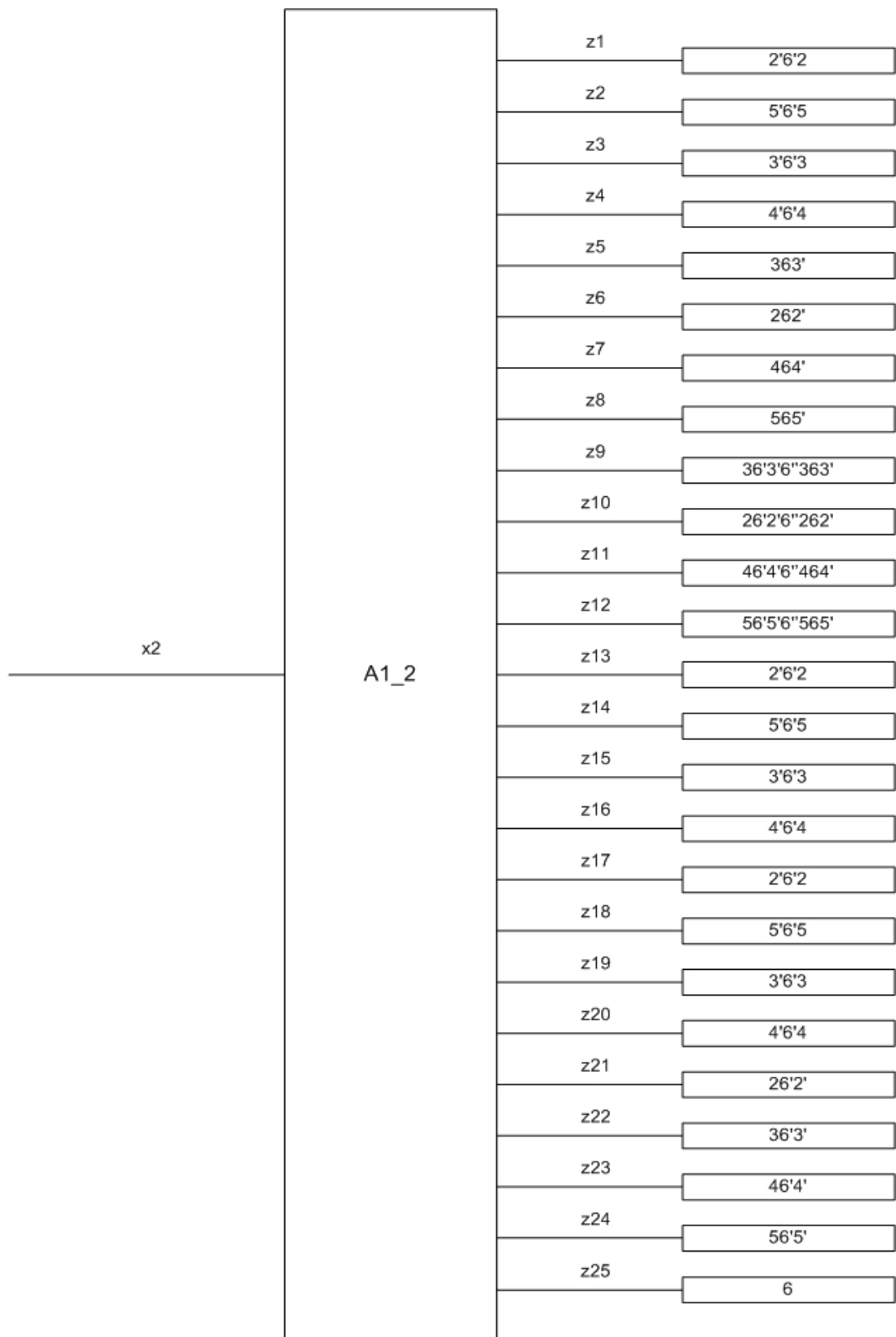


Рис. 35. Автомат 2



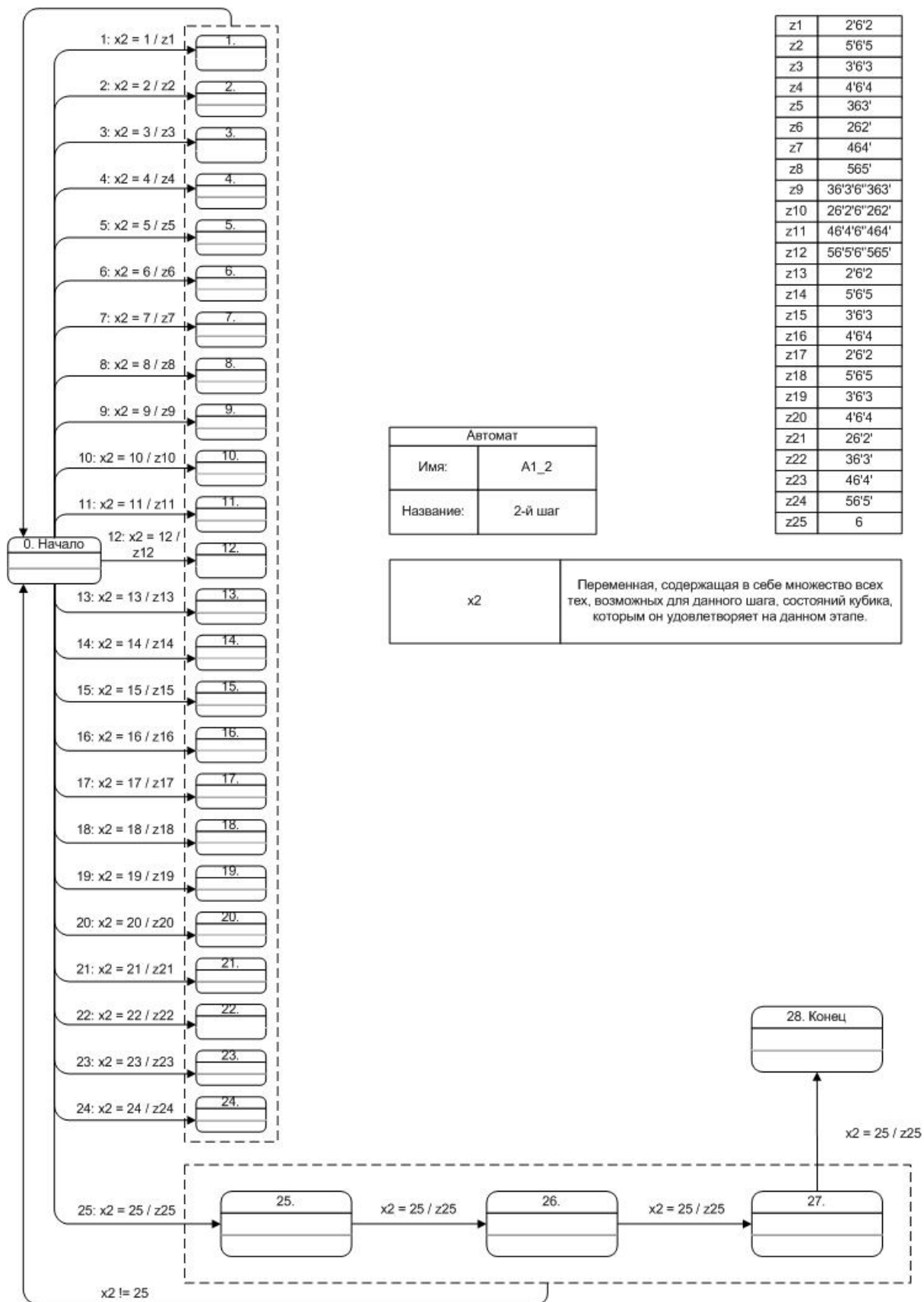


Рис. 36. Схема переходов автомата 2

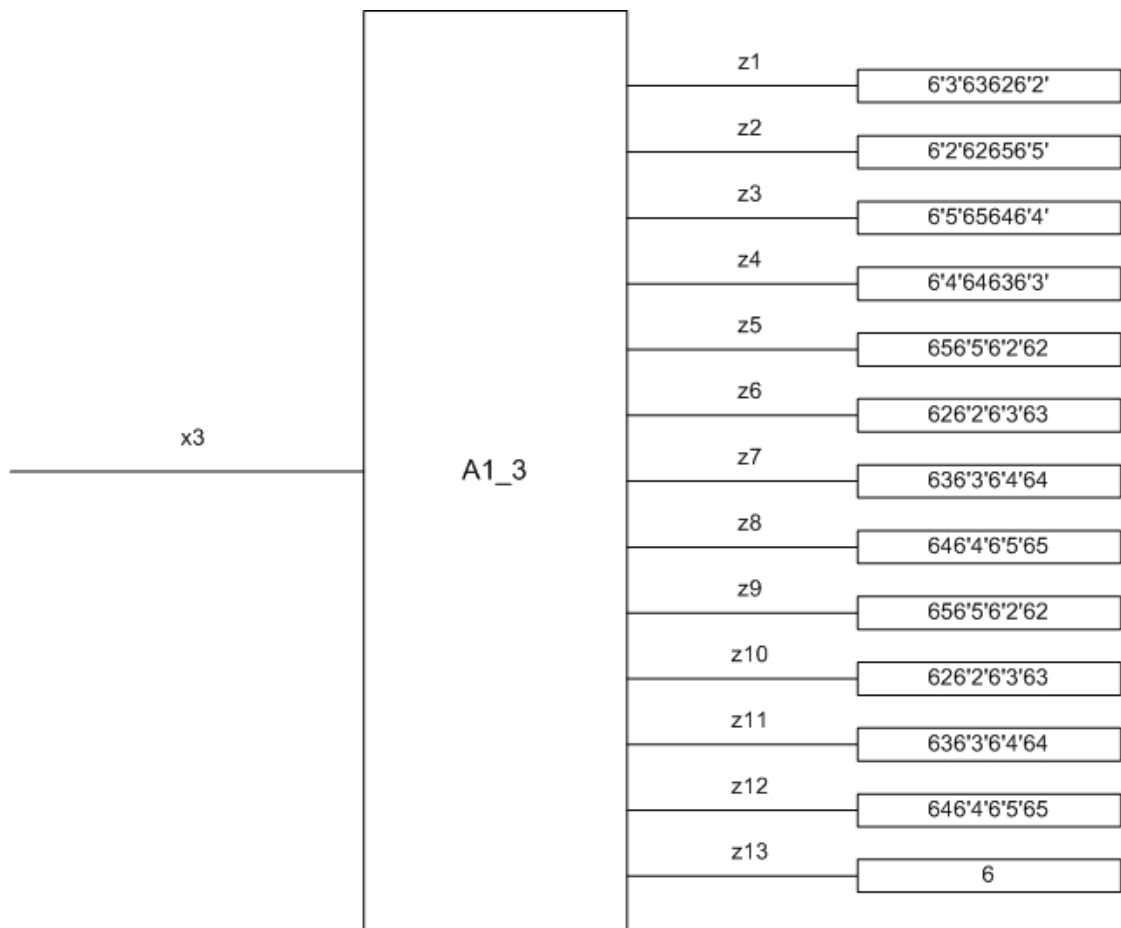


Рис. 37. Автомат 3



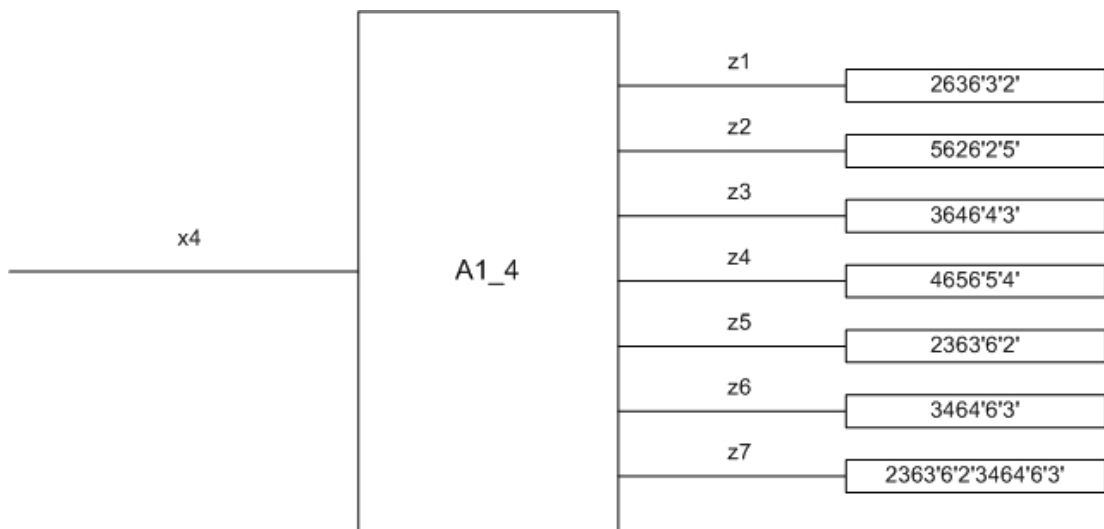
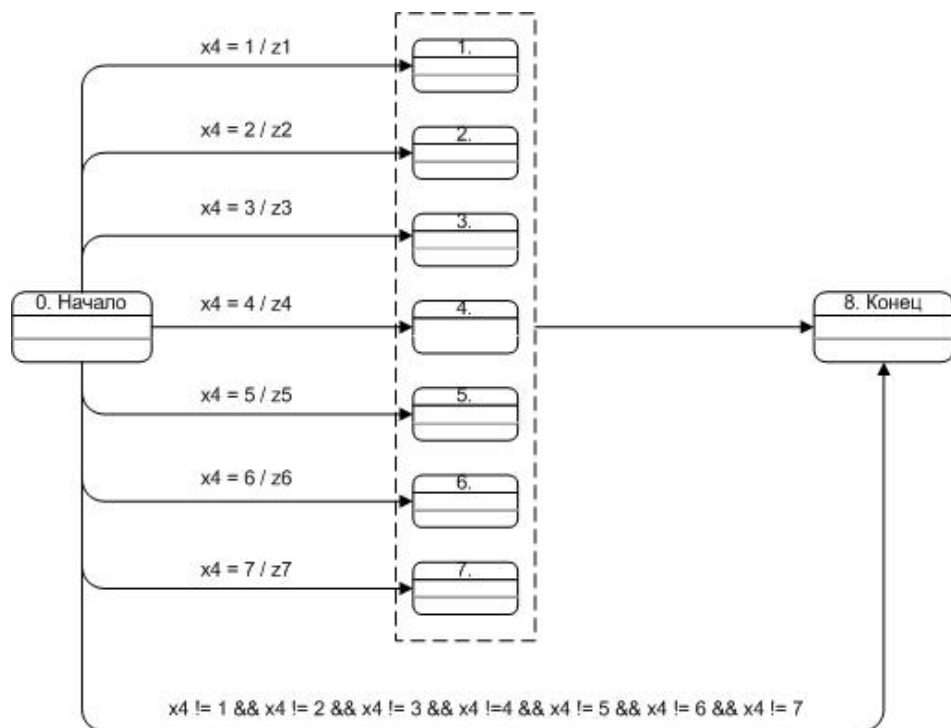


Рис. 39. Автомат 4



Автомат	
Имя:	A1_4
Название:	4-й шаг

z1	2636'3'2'
z2	5626'2'5'
z3	3646'4'3'
z4	4656'5'4'
z5	2363'6'2'
z6	3464'6'3'
z7	2363'6'2'3464'6'3'

x4	Переменная, содержащая в себе множество всех тех, возможных для данного шага, состояний кубика, которым он удовлетворяет на данном этапе.
----	---

Рис. 40. Схема переходов автомата 4

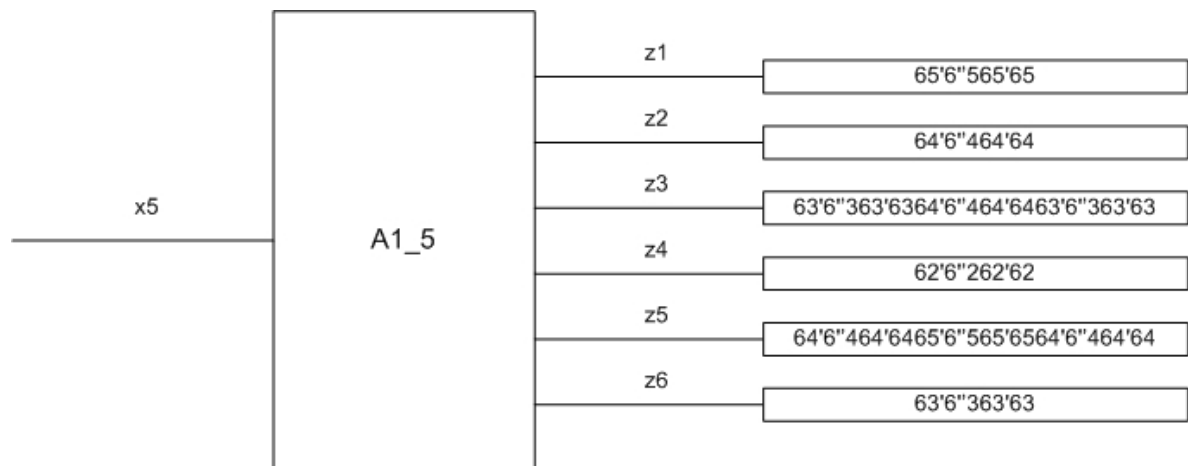
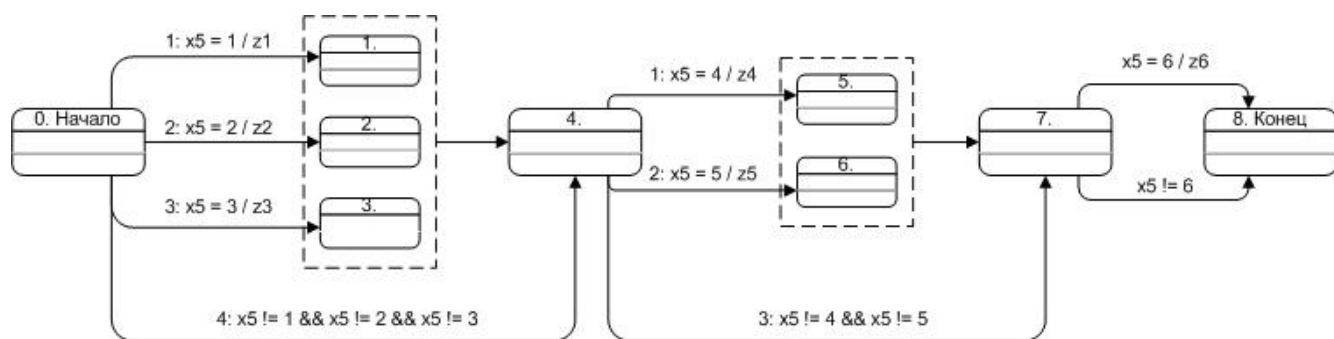


Рис. 41. Автомат 5



Автомат	
Имя:	A1_5
Название:	5-й шаг

z1	65'6''565'65
z2	64'6''464'64
z3	63'6''363'6364'6''464'6463'6''363'63
z4	62'6''262'62
z5	64'6''464'6465'6''565'6564'6''464'64
z6	63'6''363'63

x5	Переменная, содержащая в себе множество всех тех, возможных для данного шага, состояний кубика, которым он удовлетворяет на данном этапе.
----	---

Рис. 42. Схема переходов автомата 5

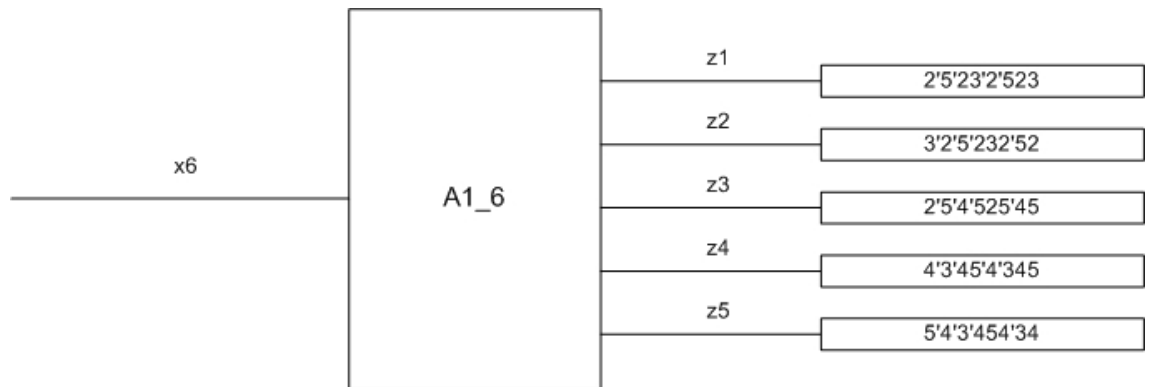
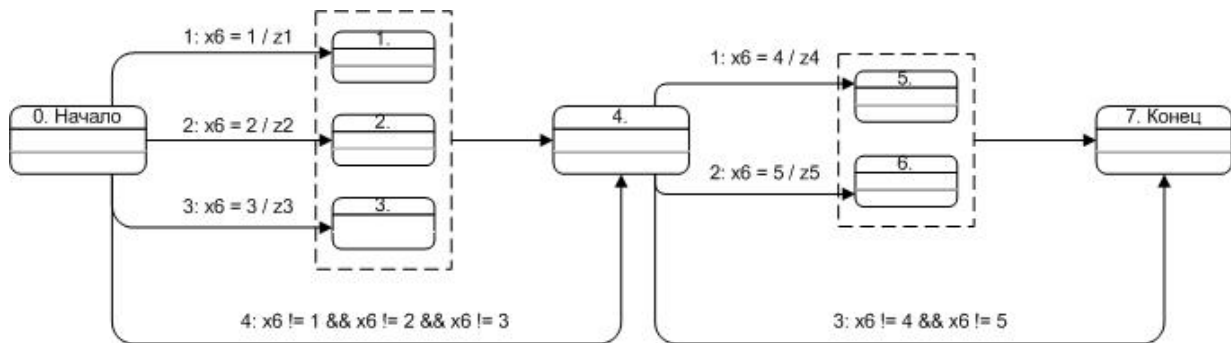


Рис. 43. Автомат 6



Автомат	
Имя:	A1_6
Название:	6-й шаг

$z_1$	2'5'23'2'523
$z_2$	3'2'5'232'52
$z_3$	2'5'4'525'45
$z_4$	4'3'45'4'345
$z_5$	5'4'3'454'34

$x_6$	Переменная, содержащая в себе множество всех тех, возможных для данного шага, состояний кубика, которым он удовлетворяет на данном этапе.
-------	---

Рис. 44. Схема переходов автомата 6

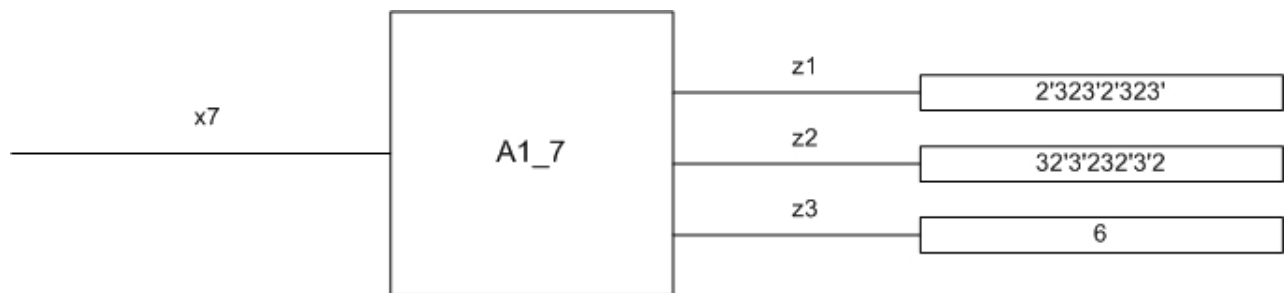


Рис. 45. Автомат 7

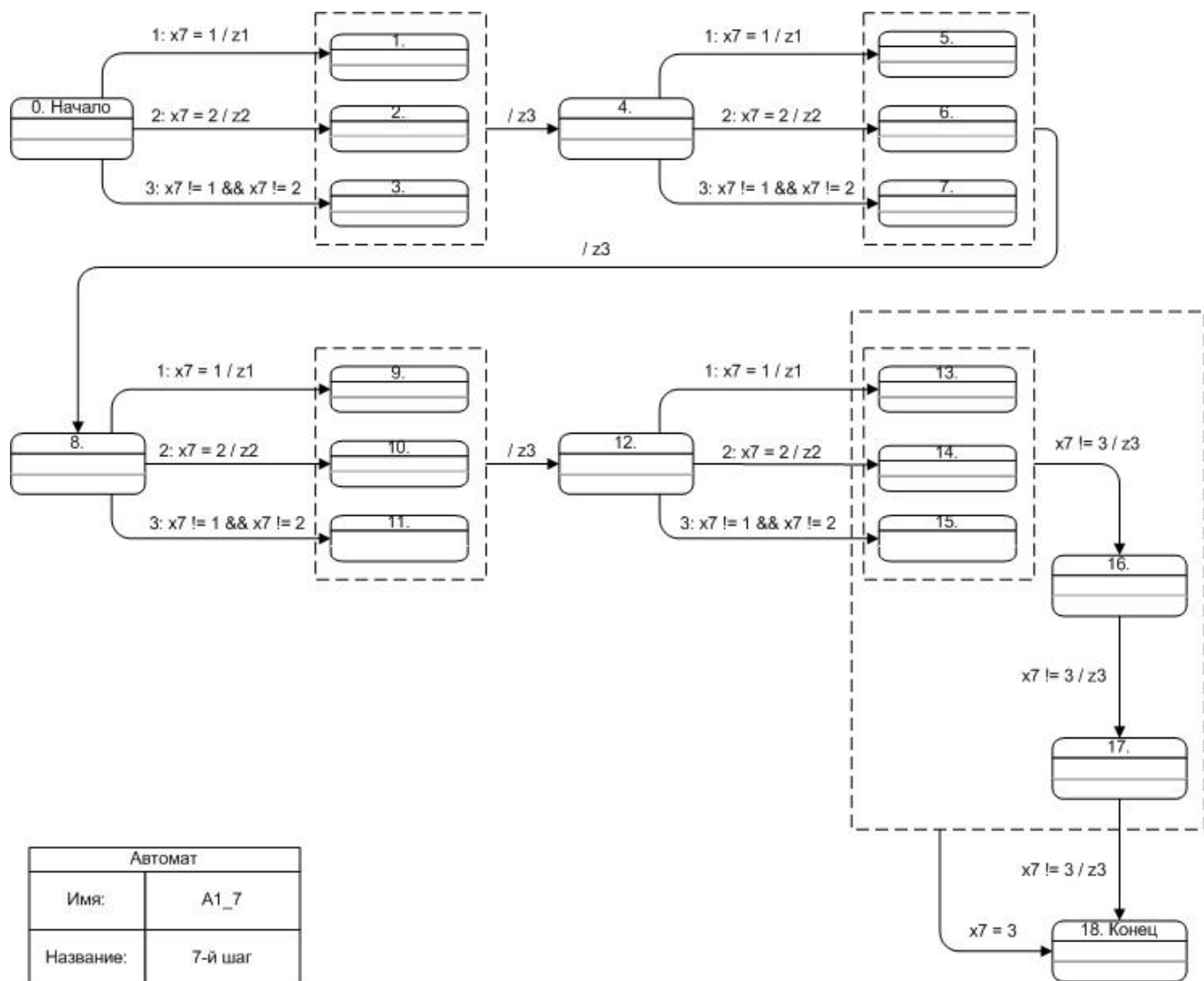


Рис. 46. Схема переходов автомата 7

## **Заключение**

Процесс разработки приложения подтвердил тот факт, что автоматное программирование целесообразно применять в тех случаях, когда для сложного алгоритма требуется формально и «прозрачно» описать его логику. Такое описание позволило создать приложение, которое стало правильно работать практически с первого раза. Вряд ли бы это удалось, если бы авторы стали писать программу традиционным путем.



## Литература

1. *Шалыто А. А., Туккель Н. И.* SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5. <http://is.ifmo.ru>, раздел «Статьи».
2. [http://www.everything2.com/index.pl?node\\_id=540195](http://www.everything2.com/index.pl?node_id=540195)
3. <http://www.ravlyk.narod.ru>
4. *Ненейвода Н. Н., Скопин И. Н.* Стили программирования. Москва-Ижевск: Институт программных систем, 2003.
5. <http://www.michaelsoft.narod.ru/kubik/kb.htm>
6. <http://www.kamlit.ru/docs/aloritms/algolist.manual.ru/misc/rubik.php.htm>
7. <http://arg.narod.ru/rubik/algo.html>

## Приложение 1. Протоколы работы программы

Все действия с кубиком отражаются в протоколе. Приняты следующие обозначения:

1. «ИНИЦИАЛИЗИРОВАН» – произошел переход в начальное состояние автомата и модели кубика в собранное состояние.
2. «ПЕРЕМЕШАН» – произошло перемешивание кубика.
3. «АВТОМАТ #...» – алгоритм перешел в автомат с указанным номером.
4. «ШАГ ...:» – автомат сделал шаг с указанным номером, далее следуют сами действия при шаге (повороты).
5. Повороты описываются номером грани:
  - один – нижняя (белая),
  - два – передняя (синяя),
  - три – левая (красная),
  - четыре – задняя (зеленая),
  - пять – правая (малиновая),
  - шесть – верхняя (желтая)).Штрих обозначает поворот против часовой стрелки на угол  $90^\circ$ , в остальных случаях подразумевается поворот по часовой стрелке на угол  $90^\circ$ .
6. «ЗАГРУЖЕН» – исходное состояние кубика загружено из файла.
7. «БЫЛО ПРОИЗВЕДЕНО:  $n$  ПОВОРОТ(ОВ)» – кубик собран,  $n$  – количество сделанных поворотов.

Пример:

ИНИЦИАЛИЗИРОВАН.

ПЕРЕМЕШАН.

АВТОМАТ #1:

ШАГ 1: 3' 6' 3

ШАГ 2: 2 2 6' 2 2

ШАГ 3: 6' 2' 5 2

ШАГ 4: 6' 5' 4 5

ШАГ 5: 3' 6' 3

ШАГ 6: 3 3 6' 3 3

ШАГ 7: 6

ШАГ 8: 6

ШАГ 9: 6' 3' 2 3

ШАГ 10: 6

ШАГ 11: 6' 4' 3 4

ШАГ 12: 6

ШАГ 13: 6

ШАГ 14: 6

ШАГ 15: 6

АВТОМАТ #2:

ШАГ 1: 3 6 3'

ШАГ 2: 2 6' 2' 6 6 2 6 2'

ШАГ 3: 4' 6' 4

ШАГ 4: 5 6' 5' 6 6 5 6 5'

.....

## Приложение 2. Исходные коды

[illegible]

```

#include "Brick0_1b.h"
#include "Brick0_1bDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

BEGIN_MESSAGE_MAP(CBrick0_1bApp, CWinApp)
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

CBrick0_1bApp::CBrick0_1bApp()
/*
    Конструктор
*/
{
}

// экземпляр CBrick0_1bApp
CBrick0_1bApp theApp;

BOOL CBrick0_1bApp::InitInstance()
/*
    Инициализация
*/
{
    InitCommonControls();

    CWinApp::InitInstance();

    AfxEnableControlContainer();
}

```

```

    CBrick0_1bDlg dlg;
    m_pMainWnd = &dlg;
    INT_PTR nResponse = dlg.DoModal();
    if (nResponse == IDOK) {
    } else if (nResponse == IDCANCEL) {
    }

    return FALSE;
}

// Brick0_1bDlg.h: описание класса CBrick0_1bDlg
//
////////////////////////////////////

#pragma once

#include <windows.h>           // Заголовочный файл для Windows
#include <gl\gl.h>             // Заголовочный файл для OpenGL32 библиотеки
#include <gl\glu.h>            // Заголовочный файл для GLu32 библиотеки
#include <gl\glaux.h>          // Заголовочный файл для GLaux библиотеки
#include "BigBrick.h"
#include "AutoBrick.h"
#include "afxwin.h"

class CBrick0_1bDlg : public CDialog
{
public:
    CBrick0_1bDlg(CWnd* pParent = NULL);
    enum { IDD = IDD_BRICK0_1B_DIALOG };

protected:
    virtual void DoDataExchange(CDataExchange* pDX); support
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

```

```

protected:
    HICON m_hIcon;

    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnTimer(UINT nIDEvent);
    DECLARE_MESSAGE_MAP()

// Поля
private:
    CDC*      pDC;
    GLfloat nXangle;
    GLfloat nYangle;
    GLfloat nZangle;

public:
    CBigBrick      br;                // экземпляр кубика
    CAutoBrick*    autobr;            // экземпляр автомата CAutoBrick
    bool           pause;              // флаг паузы
    int            step;               // номер шага
    int            delay;              // задержка
    bool           is_first;           // флаг для первого автомата

    GLvoid        InitGL();
    bool          bSetupPixelFormat();
    void          DrawScene();

    void          AddLog(int, bool);    // для добавления поворота в лог
    void          AddLog(CString);      // для добавления строки
    void          AddLog(int, int number = 0); // для добавления команды с номером
    void          EnableControls(bool, bool is_auto = false); // Дизэйбл/Раздизэйбл кнопок

public:
    afx_msg void OnBnClickedRight();

```

```

afx_msg void OnBnClickedYPos();
afx_msg void OnBnClickedXPos();
afx_msg void OnBnClickedXNeg();
afx_msg void OnNextStepBtnClick();
afx_msg void OnBnClickedAuto();
afx_msg void OnFileOpen();
afx_msg void OnFileSave();
afx_msg void OnBnClickedMixUp();
afx_msg void OnBnClickedPause();
afx_msg void OnNextMultistepBtnClick();
afx_msg void OnFileNew();

```

```
private:
```

```
    // Главный файл лога
```

```
    CEdit editLog;
```

```
public:
```

```

afx_msg void OnStapsNext();
afx_msg void OnStapsMultinext();
afx_msg void OnFileExit();
afx_msg void OnViewLogPanel();
afx_msg void OnStapsAuto();
afx_msg void OnDelay100ms();
afx_msg void OnDelay200ms();
afx_msg void OnDelay300ms();
afx_msg void OnDelay400ms();
afx_msg void OnDelay500ms();
afx_msg void OnAbout();

```

```
};
```

```
// Brick0_1bDlg.cpp: реализация класса CBrick0_1bDlg
```

```
//
```

```
////////////////////////////////////
```

```
#include "stdafx.h"
```

```
#include "Brick0_1b.h"
```

```
#include "Brick0_1bDlg.h"
```

```

#include <windows.h>           // Заголовочный файл для Windows
#include <gl\gl.h>             // Заголовочный файл для OpenGL32 библиотеки
#include <gl\glu.h>            // Заголовочный файл для GLu32 библиотеки
#include <gl\glaux.h>          // Заголовочный файл для GLaux библиотеки
#include "brick0_1bdlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

UINT BuildProc(LPVOID pParam);
UINT RandomProc(LPVOID);
UINT StepSmallProc(LPVOID pParam);
UINT StepBigProc(LPVOID pParam);

// CAboutDlg - диалоговое окно для меню "About..."
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

    enum { IDD = IDD_ABOUTBOX };

protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

protected:
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{

```



```

        CDialog::DoDataExchange(pDX);
    }

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// CBrick0_1bDlg - главное окно с визуализацией
CBrick0_1bDlg::CBrick0_1bDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CBrick0_1bDlg::IDD, pParent)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CBrick0_1bDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_EDIT1, editLog);
}

BEGIN_MESSAGE_MAP(CBrick0_1bDlg, CDialog)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    //}}AFX_MSG_MAP
    ON_BN_CLICKED(IDC_RIGHT, OnBnClickedRight)
    ON_BN_CLICKED(IDC_LEFT, OnBnClickedYPos)
    ON_BN_CLICKED(IDC_UP, OnBnClickedXPos)
    ON_BN_CLICKED(IDC_DOWN, OnBnClickedXNeg)

    ON_COMMAND(ID_FILE_NEW, OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
    ON_COMMAND(ID_FILE_SAVE, OnFileSave)
    ON_COMMAND(ID_FILE_EXIT, OnFileExit)

    ON_BN_CLICKED(IDC_MIXUP, OnBnClickedMixUp)

```

```

ON_BN_CLICKED(IDC_PAUSE, OnBnClickedPause)

ON_BN_CLICKED(IDC_STAPS_NEXT, OnNextStepBtnClick)
ON_BN_CLICKED(IDC_STAPS_MULTINEXT, OnNextMultistepBtnClick)
ON_BN_CLICKED(IDC_AUTO, OnBnClickedAuto)
ON_COMMAND(ID_STAPS_NEXT, OnStapsNext)
ON_COMMAND(ID_STAPS_MULTINEXT, OnStapsMultinext)
ON_COMMAND(ID_STAPS_AUTO, OnStapsAuto)

ON_COMMAND(ID_VIEW_LOGPANEL, OnViewLogPanel)

ON_COMMAND(ID_DELAY_100MS, OnDelay100ms)
ON_COMMAND(ID_DELAY_200MS, OnDelay200ms)
ON_COMMAND(ID_DELAY_300MS, OnDelay300ms)
ON_COMMAND(ID_DELAY_400MS, OnDelay400ms)
ON_COMMAND(ID_DELAY_500MS, OnDelay500ms)

ON_COMMAND(ID_HELP_ABOUTBRICK0, OnAbout)
END_MESSAGE_MAP()

BOOL CBrick0_1bDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    nXangle = 0;
    nYangle = 0;
    nZangle = 0;

    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
    }
}

```

```

        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    pDC = GetDC();
    CenterWindow();
    InitGL();
    SetTimer(1, 10, NULL);

    SetIcon(m_hIcon, TRUE);
    SetIcon(m_hIcon, FALSE);

    autobr = new CAutoBrick(&br); // экземпляр автомата CAutoBrick
    pause = false; // паузы нет
    step = 1; // первый шаг
    is_first = true; // флаг первого автомата
    delay = 100; // задержка 100 мс.

    CButton* btn = (CButton*)GetDlgItem(IDC_PAUSE);
    btn->SetWindowText("Пауза");
    btn = (CButton*)GetDlgItem(IDC_AUTO);
    btn->SetWindowText("Автосборка");
    btn = (CButton*)GetDlgItem(IDC_MIXUP);
    btn->SetWindowText("Перемешать");
    this->SetWindowText("Кубик Рубика v. 1.0b");

    return TRUE;
}

void CBrick0_1bDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFFF) == IDM_ABOUTBOX)
    {

```

```

        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
else
{
    CDialog::OnSysCommand(nID, lParam);
}
}

void CBrick0_1bDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this);

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        dc.DrawIcon(x, y, m_hIcon);
    } else {
        CDialog::OnPaint();
    }

    DrawScene();
}

HCURSOR CBrick0_1bDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

```

```

GLvoid CBrick0_1bDlg::InitGL()
{
    CRect rect;
    HGLRC hrc;
    if (!bSetupPixelFormat())
        return;
    hrc = wglCreateContext(pDC->GetSafeHdc());
    ASSERT(hrc != NULL);
    wglMakeCurrent(pDC->GetSafeHdc(), hrc);

    GetClientRect(&rect);
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClearDepth(1.0);
    glDepthFunc(GL_LESS);
    glEnable(GL_DEPTH_TEST);
    glShadeModel(GL_SMOOTH);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0f, (GLfloat)rect.right / (GLfloat)rect.bottom, 0.1f, 100.0f);
    glMatrixMode(GL_MODELVIEW);
    br.SetSize(0.8f);
    br.Init();
}

BOOL CBrick0_1bDlg::PreCreateWindow(CREATESTRUCT& cs)
{
    cs.style |= WS_CLIPSIBLINGS | WS_CLIPCHILDREN;
    return CDialog::PreCreateWindow(cs);
}

bool CBrick0_1bDlg::bSetupPixelFormat()
{
    static PIXELFORMATDESCRIPTOR pfd =
    {
        sizeof(PIXELFORMATDESCRIPTOR),    // размер pfd

```

```

        1, // номер версии
        PFD_DRAW_TO_WINDOW | // поддержка окон
        PFD_SUPPORT_OPENGL | // поддержка OpenGL
        PFD_DOUBLEBUFFER, // двойной буфер
        PFD_TYPE_RGBA, // RGBA тип
        24, // 24-бита для глубины цвета
        0, 0, 0, 0, 0, 0,
        0,
        0,
        0,
        0, 0, 0, 0,
        32, // 32-бита для z-буфера
        0,
        0,
        PFD_MAIN_PLANE, // главный слой
        0, // зарезервировано
        0, 0, 0
    };

    int pixelformat;

    if ((pixelformat = ChoosePixelFormat(pDC->GetSafeHdc(), &pfd)) == 0)
    {
        MessageBox("ChoosePixelFormat failed");
        return FALSE;
    }

    if (SetPixelFormat(pDC->GetSafeHdc(), pixelformat, &pfd) == FALSE)
    {
        MessageBox("SetPixelFormat failed");
        return FALSE;
    }
    return TRUE;
}

void CBrick0_1bDlg::OnTimer(UINT nIDEvent)

```

```

{
    CDialog::OnTimer(nIDEvent);
}

void CBrick0_1bDlg::DrawScene()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, -10.0f);
    glRotatef(nXangle, 1.0f, 0.0f, 0.0f);
    glRotatef(nYangle, 0.0f, 1.0f, 0.0f);
    glRotatef(nZangle, 0.0f, 0.0f, 1.0f);

    br.Draw();
    glFinish();
    SwapBuffers(wglGetCurrentDC());
}

void CBrick0_1bDlg::OnBnClickedRight()
{
    nYangle -= 10;
    DrawScene();
}

void CBrick0_1bDlg::OnBnClickedYPos()
{
    nYangle += 10;
    DrawScene();
}

void CBrick0_1bDlg::OnBnClickedXPos()
{
    nXangle += 10;
    DrawScene();
}

```

```

void CBrick0_1bDlg::OnBnClickedXNeg()
{
    nXangle -= 10;
    DrawScene();
}

// ДЕЙСТВИЯ ПО СБОРКЕ КУБИКА
void CBrick0_1bDlg::OnNextStepBtnClick()
/*
    Маленький шаг
*/
{
    if (is_first)
    {
        is_first = false;
        AddLog(1, 1);
    }
    AfxBeginThread(StepSmallProc, this);
}

void CBrick0_1bDlg::OnNextMultistepBtnClick()
/*
    Большой шаг
*/
{
    if (is_first)
    {
        is_first = false;
        AddLog(1, 1);
    }
    AfxBeginThread(StepBigProc, this);
}

void CBrick0_1bDlg::OnBnClickedAuto()

```



```

/*
    Автосборка кубика
*/
{
    if (is_first)
    {
        is_first = false;
        AddLog(1, 1);
    }
    AfxBeginThread(BuildProc, this);
}

void CBrick0_1bDlg::OnBnClickedMixUp()
/*
    Замещать
*/
{
    autobr->Reset();
    AfxBeginThread(RandomProc, this);

    is_first = true;
    step = 1;
}

void CBrick0_1bDlg::OnBnClickedPause()
/*
    Пауза
*/
{
    pause = !pause;
    CButton* btn = (CButton*)GetDlgItem(IDC_PAUSE);
    if (pause)
        btn->SetWindowText("Продолжить");
    else
        btn->SetWindowText("Пауза");
}

```

```

UINT StepSmallProc(LPVOID pParam)
/*
    Поток для маленького шага
*/
{
    CBrick0_1bDlg* pObject = (CBrick0_1bDlg*)pParam;

    // Задизэйблить кнопки
    pObject->EnableControls(false);

    // Главное тело
    list <action> act;
    int automata = pObject->autobr->automata;

    if (!pObject->autobr->finish)
    {
        act = pObject->autobr->NextStep();

        if (!act.empty())
        {
            pObject->AddLog(2, pObject->step);

            for (list <action> :: iterator i = act.begin(); i != act.end(); i++)
            {
                pObject->br.Rotate(i->side, i->direct);
                pObject->AddLog(i->side, i->direct);
                pObject->Invalidate(0);
                Sleep(pObject->delay);
            }

            pObject->step++;
            pObject->AddLog("\r\n");
        }

        if (pObject->autobr->automata != automata)
    }
}

```

```

        {
            pObject->step = 1;
            pObject->AddLog(1, pObject->autobr->automata + 1);
        }
    }

    // Раздизэйблить кнопки
    pObject->EnableControls(true);

    if (pObject->autobr->finish)
    {
        char s[128];
        sprintf(s, "Сборка завершена! (%d поворот(ов))", pObject->autobr->counter);
        pObject->MessageBox(s, "Внимание", MB_OK);
        pObject->AddLog(5);
    }
    AfxEndThread(0);

    return 0;
}

UINT StepBigProc(LPVOID pParam)
/*
    Поток для большого шага
*/
{
    CBrick0_1bDlg* pObject = (CBrick0_1bDlg*)pParam;

    // Задизэйблить кнопки
    pObject->EnableControls(false);

    // Главное тело
    list <action> act;
    int automata = pObject->autobr->automata;

    if (!pObject->autobr->finish)

```

```

{
    while (pObject->autobr->automata == automata)
    {
        act = pObject->autobr->NextStep();

        if (!act.empty())
        {
            pObject->AddLog(2, pObject->step);

            for (list <action> :: iterator i = act.begin(); i != act.end(); i++)
            {
                pObject->br.Rotate(i->side, i->direct);
                pObject->AddLog(i->side, i->direct);
                pObject->Invalidate(0);
                Sleep(pObject->delay);
            }

            pObject->step++;
            pObject->AddLog("\r\n");
        }
    }
    pObject->step = 1;
}

// Раздизэйблить кнопки
pObject->EnableControls(true);

// Завершение
if (pObject->autobr->finish) {
    char s[128];
    sprintf(s, "Сборка завершена! (%d поворот(ов))", pObject->autobr->counter);
    pObject->MessageBox(s, "Внимание", MB_OK);
    pObject->AddLog(5);
} else
    pObject->AddLog(1, pObject->autobr->automata + 1);
AfxEndThread(0);

```

```

        return 0;
    }

UINT BuildProc(LPVOID pParam)
/*
    Поток для автосборки
*/
{
    CBrick0_1bDlg* pObject = (CBrick0_1bDlg*)pParam;

    // Задизэйблить кнопки
    pObject->EnableControls(false, true);

    // Главное тело
    list <action> act;

    while (!pObject->autobr->finish)
    {
        while (pObject->pause);

        if (pObject->is_first)
        {
            pObject->AddLog(1, 1);
            pObject->is_first = false;
        }

        int automata = pObject->autobr->automata;
        act = pObject->autobr->NextStep();

        if (!act.empty())
        {
            pObject->AddLog(2, pObject->step);

            for (list <action> :: iterator i = act.begin(); i != act.end(); i++)
            {

```

```

        pObject->br.Rotate(i->side, i->direct);
        pObject->AddLog(i->side, i->direct);
        pObject->Invalidate(0);
        Sleep(pObject->delay);
    }

    pObject->step++;
    pObject->AddLog("\r\n");
}

if (pObject->autobr->automata != automata)
{
    if (pObject->autobr->automata < 7)
        pObject->AddLog(1, pObject->autobr->automata + 1);
    pObject->step = 1;
}
}

// Раздизэйблить кнопки
pObject->EnableControls(true, true);

// Завершение
char s[128];
sprintf(s, "Сборка завершена! (%d поворот(ов))", pObject->autobr->counter);
pObject->MessageBox(s, "Внимание", MB_OK);
pObject->AddLog(5);
AfxEndThread(0);

return 0;
}

UINT RandomProc(LPVOID pParam)
/*
    Поток для размешивания
*/

```

```

{
    CBrick0_1bDlg* pObject = (CBrick0_1bDlg*)pParam;

    // Задизэйблить кнопки
    pObject->EnableControls(false);

    // Главное тело
    srand((unsigned)time(NULL));

    int iter = rand() % 20 + 30;
    for (int i = 0; i < iter; i++)
    {
        int side = rand() % 6 + 1;
        bool direct;
        if (rand() % 2)
            direct = true;
        else
            direct = false;

        pObject->br.Rotate(side, direct);
        pObject->Invalidate(0);
        Sleep(pObject->delay / 20);
    }
    pObject->AddLog("Перемешан.\r\n");

    // Раздизэйблить кнопки
    pObject->EnableControls(true);

    // Завершение
    AfxEndThread(0);

    return 0;
}

// "МЕНЮ"

```

```

void CBrick0_1bDlg::OnFileNew()
{
    autobr->Reset();
    is_first = true;
    br.Init();
    editLog.SetWindowText("");

    DrawScene();

    AddLog("Инициализирован.\r\n");
}

void CBrick0_1bDlg::OnFileOpen()
/*
    Метод отвечает за открытие и считывание и считывание конфигурации кубика из файла.
    Структуру файла см метод OnFileSave()
*/
{
    CString sPathName;
    CFileDialog *dlg = new CFileDialog(true, "brk", ".brk", OFN_FILEMUSTEXIST|
OFN_HIDEREADONLY,
                                "Brick Files (*.brk)|*.brk||", this, NULL);
    if(dlg->DoModal() == IDOK)
    {
        sPathName = dlg->GetPathName();
        CFile file;

        autobr->Reset();
        is_first = true;
        br.Init();

        try
        {
            file.Open(sPathName, CFile::modeRead , NULL);
            for(int i = 0; i < 3; i++)
            {

```



```

        for(int j = 0; j < 3; j++)
        {
            for(int k = 0; k < 3; k++)
            {
                char * ch = new char;
                for(int h = 0; h < 6; h++)
                {
                    file.Read(ch,1);
                    br.SetColor(i, j, k, h, atoi(ch));
                }
            }
        }
        DrawScene();

        AddLog("Файл загружен.\r\n");
    }
    catch(CFileException *e)
    {
        e->Delete();
        AfxMessageBox("Ошибка при открытии .", MB_OK | MB_ICONERROR, NULL);
    }
}
delete dlg;
}

void CBrick0_1bDlg::OnFileSave()
/*
    Метод отвечает за запись текущей конфигурации кубика в файл
    Файл с расширением ".brk"
    Структура файла: поочередно записываются цвета граней всех маленьких кубиков
*/
{
    CString sPathName;
    CFileDialog *dlg = new CFileDialog(false, "brk", ".brk", OFN_OVERWRITEPROMPT ,

```

```

        "Brick Files (*.brk)|*.brk||", this, NULL);
if(dlg->DoModal() == IDOK)
{
    sPathName = dlg->GetPathName();
    CFile file;
    try
    {
        file.Open(sPathName,CFile::modeCreate | CFile::modeWrite, NULL);
        for(int i = 0; i < 3; i++)
        {
            for(int j = 0; j < 3; j++)
            {
                for(int k = 0; k < 3; k++)
                {
                    char *ch = new char;
                    for(int h = 0; h < 6; h++)
                    {
                        itoa(br.GetColor(i, j, k, h),ch,10);
                        file.Write(ch, 1);
                    }
                }
            }
        }
    }
    catch(CFileException *e)
    {
        e->Delete();
        AfxMessageBox("Ошибка при записи.", MB_OK | MB_ICONERROR, NULL);
    }
}
delete dlg;

}

void CBrick0_1bDlg::OnFileExit()
/*

```

```

        Выход
    */
    {
        exit(0);
    }

void CBrick0_1bDlg::OnStapsNext()
/*
    Маленький шаг
*/
{
    OnNextStepBtnClick();
}

void CBrick0_1bDlg::OnStapsMultinext()
/*
    Большой шаг
*/
{
    OnNextMultistepBtnClick();
}

void CBrick0_1bDlg::OnStapsAuto()
/*
    Автосборка
*/
{
    OnBnClickedAuto();
}

void CBrick0_1bDlg::OnDelay100ms()
/*
    Выбор задержки
*/
{
    CMenu* menu = (CMenu*)GetMenu();

```

```

    menu->CheckMenuItem(ID_DELAY_100MS, MF_CHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_200MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_300MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_400MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_500MS, MF_UNCHECKED | MF_BYCOMMAND);

    delay = 100;
}

void CBrick0_1bDlg::OnDelay200ms()
/*
    Выбор задержки
*/
{
    CMenu* menu = (CMenu*)GetMenu();
    menu->CheckMenuItem(ID_DELAY_100MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_200MS, MF_CHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_300MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_400MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_500MS, MF_UNCHECKED | MF_BYCOMMAND);

    delay = 200;
}

void CBrick0_1bDlg::OnDelay300ms()
/*
    Выбор задержки
*/
{
    CMenu* menu = (CMenu*)GetMenu();
    menu->CheckMenuItem(ID_DELAY_100MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_200MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_300MS, MF_CHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_400MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_500MS, MF_UNCHECKED | MF_BYCOMMAND);
}

```

```

        delay = 300;
    }

void CBrick0_1bDlg::OnDelay400ms()
/*
    Выбор задержки
*/
{
    CMenu* menu = (CMenu*)GetMenu();
    menu->CheckMenuItem(ID_DELAY_100MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_200MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_300MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_400MS, MF_CHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_500MS, MF_UNCHECKED | MF_BYCOMMAND);

    delay = 400;
}

void CBrick0_1bDlg::OnDelay500ms()
/*
    Выбор задержки
*/
{
    CMenu* menu = (CMenu*)GetMenu();
    menu->CheckMenuItem(ID_DELAY_100MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_200MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_300MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_400MS, MF_UNCHECKED | MF_BYCOMMAND);
    menu->CheckMenuItem(ID_DELAY_500MS, MF_CHECKED | MF_BYCOMMAND);

    delay = 500;
}

void CBrick0_1bDlg::OnViewLogPanel()
/*
    Спрятать/Показать панель лога

```

```

*/
{
    CMenu* menu = (CMenu*)GetMenu();
    UINT flags;
    flags = menu->GetMenuState(ID_VIEW_LOGPANEL, MF_BYCOMMAND);

    if (flags & MF_CHECKED) {
        menu->CheckMenuItem(ID_VIEW_LOGPANEL, MF_UNCHECKED | MF_BYCOMMAND);

        editLog.ShowWindow(SW_HIDE);
    } else {
        menu->CheckMenuItem(ID_VIEW_LOGPANEL, MF_CHECKED | MF_BYCOMMAND);

        editLog.ShowWindow(SW_SHOW);
    }
}

void CBrick0_1bDlg::OnAbout()
/*
    Меню "About..."
*/
{
    CAboutDlg dlgAbout;
    dlgAbout.DoModal();
}

// ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ
void CBrick0_1bDlg::AddLog(int side, bool dir)
/*
    Добавить поворот в лог
*/
{
    CString str;
    editLog.GetWindowText(str);

    switch (side)

```

```

{
case 1:                                // Низ
    if (dir)
        str += "1 ";
    else
        str += "1' ";
    break;
case 2:                                // Передняя
    if (dir)
        str += "2 ";
    else
        str += "2' ";
    break;
case 3:                                // Правая
    if (dir)
        str += "3 ";
    else
        str += "3' ";
    break;
case 4:                                // Задняя
    if (dir)
        str += "4 ";
    else
        str += "4' ";

    break;
case 5:                                // Левая
    if (dir)
        str += "5 ";
    else
        str += "5' ";
    break;
case 6:                                // Верхняя
    if (dir)
        str += "6 ";
    else

```

```

        str += "6' ";
        break;
    }

    editLog.SetWindowText(str);
}

void CBrick0_1bDlg::AddLog(CString newStr)
/*
    Добавить строку в лог
*/
{
    CString str;
    editLog.GetWindowText(str);

    str += newStr;
    editLog.SetWindowText(str);
}

void CBrick0_1bDlg::AddLog(int command, int number)
/*
    Записать в лог команду с номером
*/
{
    char temp[16];
    CString str;
    editLog.GetWindowText(str);

    switch (command)
    {
    case 1:
        str += "АВТОМАТ #";
        str += itoa(number, temp, 10);
        str += ":\r\n";
        break;
    case 2:

```



```

        str += "Шар ";
        str += itoa(number, temp, 10);
        str += ": ";
        break;
case 3:
    str += "Перемешан.\r\n";
    break;
case 4:
    str += "Инициализирован.\r\n";
    break;
case 5:
    str += "Было произведено: ";
    str += itoa(autobr->counter, temp, 10);
    str += " поворот(ов).\r\n";
}

editLog.SetWindowText(str);
}

void CBrick0_1bDlg::EnableControls(bool enable, bool is_auto)
{
    CButton* st = (CButton*)GetDlgItem(IDC_STAPS_NEXT);
    CButton* mu = (CButton*)GetDlgItem(IDC_STAPS_MULTINEXT);
    CButton* au = (CButton*)GetDlgItem(IDC_AUTO);
    CButton* pa = (CButton*)GetDlgItem(IDC_PAUSE);
    CButton* mi = (CButton*)GetDlgItem(IDC_MIXUP);
    CMenu* menu = GetMenu();
    CMenu* submenu;

    if (!enable) {
        st->EnableWindow(false);
        mu->EnableWindow(false);
        au->EnableWindow(false);
        if (!is_auto) pa->EnableWindow(false);
        mi->EnableWindow(false);
    }
}

```

```

        submenu = menu->GetSubMenu(0);
        submenu->EnableMenuItem(ID_FILE_NEW, MF_BYCOMMAND | MF_DISABLED | MF_GRAYED);
        submenu->EnableMenuItem(ID_FILE_OPEN, MF_BYCOMMAND | MF_DISABLED | MF_GRAYED);
        submenu->EnableMenuItem(ID_FILE_SAVE, MF_BYCOMMAND | MF_DISABLED | MF_GRAYED);
        submenu = menu->GetSubMenu(1);
        submenu->EnableMenuItem(ID_STAPS_NEXT, MF_BYCOMMAND | MF_DISABLED | MF_GRAYED);
        submenu->EnableMenuItem(ID_STAPS_MULTINEXT, MF_BYCOMMAND | MF_DISABLED | MF_GRAYED);
    } else {
        st->EnableWindow(true);
        mu->EnableWindow(true);
        au->EnableWindow(true);
        if (!is_auto) pa->EnableWindow(true);
        mi->EnableWindow(true);

        submenu = menu->GetSubMenu(0);
        submenu->EnableMenuItem(ID_FILE_NEW, MF_BYCOMMAND | MF_ENABLED);
        submenu->EnableMenuItem(ID_FILE_OPEN, MF_BYCOMMAND | MF_ENABLED);
        submenu->EnableMenuItem(ID_FILE_SAVE, MF_BYCOMMAND | MF_ENABLED);
        submenu = menu->GetSubMenu(1);
        submenu->EnableMenuItem(ID_STAPS_NEXT, MF_BYCOMMAND | MF_ENABLED);
        submenu->EnableMenuItem(ID_STAPS_MULTINEXT, MF_BYCOMMAND | MF_ENABLED);
    }
}

```

```

// AutoBrick.h: описание класса CAutoBrick

```

```

//

```

```

////////////////////////////////////

```

```

#include "BigBrick.h"

```

```

#include <list>

```

```

using namespace std;

```

```

// Действие - единичный поворот одной из граней

```

```

struct action
{
    int    side;
    bool   direct;
};

// Автомат по сборке кубика
class CAutoBrick
{
private:
    CBigBrick*    br;                // ссылка на экземпляр кубика
    int           state;             // номер состояния
public:
    bool          finish;            // флаг завершения автомата
    int           automata;          // номер текущего автомата
    long          counter;           // количество сделанных поворотов

public:
    CAutoBrick();                    // конструкторы
    CAutoBrick(CBigBrick*);
    virtual ~CAutoBrick(void);       // деструктор

    void Reset();                    // сброс автомата
    list <action> NextStep();         // следующий шаг

private:
    // Условия переходов в соответствующем автомате
    list <int> x1();
    list <int> x2();
    list <int> x3();
    list <int> x4();
    list <int> x5();
    list <int> x6();
    list <int> x7();

```

```

// Действия при переходе
list <action> z1();
list <action> z2();
list <action> z3();
list <action> z4();
list <action> z5();
list <action> z6();
list <action> z7();
list <action> z8();
list <action> z9();
list <action> z10();
list <action> z11();
list <action> z12();
list <action> z13();
list <action> z14();
list <action> z15();
list <action> z16();
list <action> z17();
list <action> z18();
list <action> z19();
list <action> z20();
list <action> z21();
list <action> z22();
list <action> z23();
list <action> z24();
list <action> z25();

list <action> z(int);

// Вспомогательные методы
void AddAction(list <action> *, int, bool);
bool IsIn(int, list <int>);
bool IsEqual(int, int, int, int, int, int);
};

```

```

// AutoBrick.cpp: реализация класса CAutoBrick
//
////////////////////////////////////

#include "stdafx.h"
#include "AutoBrick.h"

CAutoBrick::CAutoBrick()
{
}

CAutoBrick::CAutoBrick(CBigBrick* br)
{
    this->br = br;

    state = 0;
    automata = 0;
    finish = false;
    counter = 0;
}

CAutoBrick::~CAutoBrick()
{
}

void CAutoBrick::Reset()
/*
    Сброс автомата
*/
{
    state = 0;
    automata = 0;
    finish = false;
    counter = 0;
}

```

```

list <action> CAutoBrick::NextStep()
/*
    Главный метод для получения информации о текущих шагах
*/
{
    list <action> empty;

    if (automata == 0)
    {
        // Шаг №1 - автомат #1
        switch (state)
        {
            case 0:
                for (int i = 1; i <= 25; i++)
                {
                    if (IsIn(i, x1())) // входит ли состояние i в мно-во всех состояний
                    {
                        state = i;
                        counter += z(i).size();
                        return z(i);
                    }
                }
                break;
            case 1:          case 2:          case 3:          case 4:          case 5:          case 6:
            case 7:          case 8:          case 9:          case 10:         case 11:         case 12:
            case 13:         case 14:         case 15:         case 16:         case 17:         case 18:
            case 19:         case 20:         case 21:         case 22:         case 23:         case 24:
                state = 0;
                break;
            case 25:
                if (IsIn(25, x1())) { // входит ли состояние 25 в мно-во всех состояний
                    state = 26;
                    counter += z25().size();
                    return z25();
                } else {
                    state = 0;
                }
        }
    }
}

```

```

        break;
    case 26:
        if (IsIn(25, x1())) {           // входит ли состояние 25 в мно-во всех состояний
            state = 27;
            counter += z25().size();
            return z25();
        } else {
            state = 0;
        }
        break;
    case 27:
        if (IsIn(25, x1())) {           // входит ли состояние 25 в мно-во всех состояний
            state = 28;
            counter += z25().size();
            return z25();
        } else {
            state = 0;
        }
        break;
    case 28:
        state = 0;
        automata = 1;                   // к следующему автомату
        break;
    default:
        break;
}
} else if (automata == 1) {
    // Шаг №2 - автомат #2
    switch (state)
    {
        case 0:
            for (int i = 1; i <= 25; i++)
            {
                if (IsIn(i, x2()))       // входит ли состояние i в мно-во всех состояний
                {
                    state = i;
                    counter += z(i).size();
                    return z(i);
                }
            }
        }
    }
}

```

```

    }
    break;
case 1:      case 2:      case 3:      case 4:      case 5:      case 6:
case 7:      case 8:      case 9:      case 10:     case 11:     case 12:
case 13:     case 14:     case 15:     case 16:     case 17:     case 18:
case 19:     case 20:     case 21:     case 22:     case 23:     case 24:
    state = 0;
    break;
case 25:
    if (IsIn(25, x2())) {                // входит ли состояние 25 в мно-во всех состояний
        state = 26;
        counter += z25().size();
        return z25();
    } else {
        state = 0;
    }
    break;
case 26:
    if (IsIn(25, x2())) {                // входит ли состояние 25 в мно-во всех состояний
        state = 27;
        counter += z25().size();
        return z25();
    } else {
        state = 0;
    }
    break;
case 27:
    if (IsIn(25, x2())) {                // входит ли состояние 25 в мно-во всех состояний
        state = 28;
        counter += z25().size();
        return z25();
    } else {
        state = 0;
    }
    break;
case 28:
    state = 0;
    automata = 2;                        // к следующему автомату
    break;

```



```

        default:
            break;
    }
} else if (automata == 2) {
    // Шаг №3 - автомат #3
    switch (state)
    {
        case 0:
            for (int i = 1; i <= 8; i++)
            {
                if (IsIn(i, x3())) // входит ли состояние i в мно-во всех состояний
                {
                    state = i;
                    counter += z(i).size();
                    return z(i);
                }
            }

            if (IsIn(13, x3())) // входит ли состояние 13 в мно-во всех состояний
            {
                state = 9;
                counter += z13().size();
                return z13();
            }
            break;
        case 1:          case 2:          case 3:          case 4:
        case 5:          case 6:          case 7:          case 8:
            state = 0;
            break;
        case 9:
            if (IsIn(13, x3())) // входит ли состояние 13 в мно-во всех состояний
            {
                state = 10;
                counter += z13().size();
                return z13();
            }

            for (int i = 1; i <= 8; i++)
            {

```

```

        if (IsIn(i, x3()))                // входит ли состояние i в мно-во всех состояний
        {
            state = 0;
            break;
        }
    }
    break;
case 10:
    if (IsIn(13, x3()))                    // входит ли состояние 13 в мно-во всех состояний
    {
        state = 11;
        counter += z13().size();
        return z13();
    }

    for (int i = 1; i <= 8; i++)
    {
        if (IsIn(i, x3()))                // входит ли состояние i в мно-во всех состояний
        {
            state = 0;
            break;
        }
    }
    break;
case 11:
    if (IsIn(13, x3()))                    // входит ли состояние 13 в мно-во всех состояний
    {
        state = 12;
        counter += z13().size();
        return z13();
    }

    for (int i = 1; i <= 8; i++)
    {
        if (IsIn(i, x3()))                // входит ли состояние i в мно-во всех состояний
        {
            state = 0;
            break;
        }
    }

```

```

    }
    break;
case 12:
    for (int i = 9; i <= 12; i++)
    {
        if (IsIn(i, x3())) // входит ли состояние i в мно-во всех состояний
        {
            state = i + 4;
            counter += z(i).size();
            return z(i);
        }
    }

    state = 17;
    break;
case 13: case 14: case 15: case 16:
    state = 0;
    break;
case 17:
    state = 0;
    automata = 3; // к следующему автомату
    break;
default:
    break;
}
} else if (automata == 3) {
    // Шаг №4 - автомат #4
    switch (state)
    {
        case 0:
            for (int i = 1; i <= 7; i++)
            {
                if (IsIn(i, x4())) // входит ли состояние i в мно-во всех состояний
                {
                    state = i;
                    counter += z(i).size();
                    return z(i);
                }
            }
        }
    }
}

```

```

        state = 8;
        break;
    case 1:          case 2:          case 3:          case 4:
    case 5:          case 6:          case 7:
        state = 8;
        break;
    case 8:
        state = 0;
        automata = 4;           // к следующему автомату
        break;
    default:
        break;
}
} else if (automata == 4) {
    // Шаг №5 - автомат #5
    switch (state)
    {
        case 0:
            for (int i = 1; i <= 3; i++)
            {
                if (IsIn(i, x5()))           // входит ли состояние i в мно-во всех состояний
                {
                    state = i;
                    counter += z(i).size();
                    return z(i);
                }
            }

            state = 4;
            break;
        case 1:          case 2:          case 3:
            state = 4;
            break;
        case 4:
            for (int i = 4; i <= 5; i++)
            {
                if (IsIn(i, x5()))           // входит ли состояние i в мно-во всех состояний
                {

```

```

        state = i + 1;
        counter += z(i).size();
        return z(i);
    }
}

state = 7;
break;
case 5:          case 6:
    state = 7;
    break;
case 7:
    if (IsIn(6, x5())) {                // входит ли состояние 6 в мно-во всех состояний
        state = 8;
        counter += z6().size();
        return z6();
    } else {
        state = 8;
    }
    break;
case 8:
    state = 0;
    automata = 5;                       // к следующему автомату
    break;
default:
    break;
}
} else if (automata == 5) {
    // Шаг №6 - автомат #6
    switch (state)
    {
        case 0:
            for (int i = 1; i <= 3; i++)
            {
                if (IsIn(i, x6()))        // входит ли состояние i в мно-во всех состояний
                {
                    state = i;
                    counter += z(i).size();
                }
            }
        }
    }
}

```

```

        return z(i);
    }
}

state = 4;
break;
case 1:      case 2:      case 3:
state = 4;
break;
case 4:
for (int i = 4; i <= 5; i++)
{
    if (IsIn(i, x6()))          // ВХОДИТ ЛИ СОСТОЯНИЕ i В МНО-ВО ВСЕХ СОСТОЯНИЙ
    {
        state = i + 1;
        counter += z(i).size();
        return z(i);
    }
}

state = 7;
break;
case 5:      case 6:
state = 7;
break;
case 7:
state = 0;
automata = 6;          // к следующему автомату
break;
default:
break;
}
} else if (automata == 6) {
    // Шаг №7 - автомат #7
    switch (state)
    {
        case 0:
            for (int i = 1; i <= 2; i++)
            {

```

```

        if (IsIn(i, x7()))          // входит ли состояние i в мно-во всех состояний
        {
            state = i;
            counter += z(i).size();
            return z(i);
        }
    }

    state = 3;
    break;
case 1:          case 2:          case 3:
    state = 4;
    counter += z3().size();
    return z3();
    break;
case 4:
    for (int i = 1; i <= 2; i++)
    {
        if (IsIn(i, x7()))          // входит ли состояние i в мно-во всех состояний
        {
            state = i + 4;
            counter += z(i).size();
            return z(i);
        }
    }

    state = 7;
    break;
case 5:          case 6:          case 7:
    state = 8;
    return z3();
    break;
case 8:
    for (int i = 1; i <= 2; i++)
    {
        if (IsIn(i, x7()))          // входит ли состояние i в мно-во всех состояний
        {
            state = i + 8;
            counter += z(i).size();

```

```

        return z(i);
    }
}

state = 11;
break;
case 9:      case 10:  case 11:
    state = 12;
    counter += z3().size();
    return z3();
    break;
case 12:
    for (int i = 1; i <= 2; i++)
    {
        if (IsIn(i, x7()))          // входит ли состояние i в мно-во всех состояний
        {
            state = i + 12;
            counter += z(i).size();
            return z(i);
        }
    }

    state = 15;
    break;
case 13: case 14: case 15:
    if (IsIn(3, x7())) {          // входит ли состояние 3 в мно-во всех состояний
        state = 18;
    } else {
        state = 16;
        counter += z3().size();
        return z3();
    }
    break;
case 16:
    if (IsIn(3, x7())) {          // входит ли состояние 3 в мно-во всех состояний
        state = 18;
    } else {
        state = 17;
        counter += z3().size();
    }
}

```



```

        return z3();
    }
    break;
case 17:
    if (IsIn(3, x7())) { // входит ли состояние 3 в мно-во всех состояний
        state = 18;
    } else {
        state = 18;
        counter += z3().size();
        return z3();
    }
    break;
case 18:
    automata = 7; // конец автомата
    finish = true; // флаг завершения
    break;
default:
    break;
}
}

return empty;
}

```

```

// Условия переходов
list <int> private CAutoBrick::x1()
/*
    Метод для выяснения условий перехода
*/
{
    list <int> st;

    if ( (br->GetColor(2, 2) == 2) && (br->GetColor(6, 8) == 1) ) // I.a.1
        st.push_back(1);

    if ( (br->GetColor(3, 2) == 3) && (br->GetColor(6, 6) == 1) ) // I.a.2
        st.push_back(2);
}

```

```

if ( (br->GetColor(5, 2) == 5) && (br->GetColor(6, 4) == 1) )           // I.a.3
    st.push_back(3);

if ( (br->GetColor(4, 2) == 4) && (br->GetColor(6, 2) == 1) )           // I.a.4
    st.push_back(4);

if ( (br->GetColor(2, 2) == 1) && (br->GetColor(6, 8) == 2) )           // I.b.1
    st.push_back(5);

if ( (br->GetColor(3, 2) == 1) && (br->GetColor(6, 6) == 3) )           // I.b.2
    st.push_back(6);

if ( (br->GetColor(5, 2) == 1) && (br->GetColor(6, 4) == 5) )           // I.b.3
    st.push_back(7);

if ( (br->GetColor(4, 2) == 1) && (br->GetColor(6, 2) == 4) )           // I.b.4
    st.push_back(8);

if (br->GetColor(2, 6) == 1)                                           // I.c.1
    st.push_back(9);

if (br->GetColor(3, 6) == 1)                                           // I.c.2
    st.push_back(10);

if (br->GetColor(5, 6) == 1)                                           // I.c.3
    st.push_back(11);

if (br->GetColor(4, 6) == 1)                                           // I.c.4
    st.push_back(12);

if (br->GetColor(2, 4) == 1)                                           // I.d.1
    st.push_back(13);

if (br->GetColor(3, 4) == 1)                                           // I.d.2
    st.push_back(14);

```

```

if (br->GetColor(5, 4) == 1)                                // I.d.3
    st.push_back(15);

if (br->GetColor(4, 4) == 1)                                // I.d.4
    st.push_back(16);

if (br->GetColor(2, 8) == 1)                                // I.e.1
    st.push_back(17);

if (br->GetColor(3, 8) == 1)                                // I.e.2
    st.push_back(18);

if (br->GetColor(5, 8) == 1)                                // I.e.3
    st.push_back(19);

if (br->GetColor(4, 8) == 1)                                // I.e.4
    st.push_back(20);

if ( (br->GetColor(2, 8) != 2) && (br->GetColor(1, 2) == 1) ) // I.f.1
    st.push_back(21);

if ( (br->GetColor(3, 8) != 3) && (br->GetColor(1, 6) == 1) ) // I.f.2
    st.push_back(22);

if ( (br->GetColor(5, 8) != 5) && (br->GetColor(1, 4) == 1) ) // I.f.3
    st.push_back(23);

if ( (br->GetColor(4, 8) != 4) && (br->GetColor(1, 8) == 1) ) // I.f.4
    st.push_back(24);

if (st.empty())                                            // Если ни один из тех, что выше
    st.push_back(25);

return st;

```

```

}

list <int> private CAutoBrick::x2()
/*
    Метод для выяснения условий перехода
*/
{
    list <int> st;

    if ( (br->GetColor(2, 3) == 1) && (br->GetColor(3, 1) == 3) && (br->GetColor(6, 9) == 2) )
        // II.a.1
        st.push_back(1);

    if ( (br->GetColor(5, 3) == 1) && (br->GetColor(2, 1) == 2) && (br->GetColor(6, 7) == 5) )
        // II.a.2
        st.push_back(2);

    if ( (br->GetColor(3, 3) == 1) && (br->GetColor(4, 1) == 4) && (br->GetColor(6, 3) == 3) )
        // II.a.3
        st.push_back(3);

    if ( (br->GetColor(4, 3) == 1) && (br->GetColor(5, 1) == 5) && (br->GetColor(6, 1) == 4) )
        // II.a.4
        st.push_back(4);

    if ( (br->GetColor(2, 3) == 2) && (br->GetColor(3, 1) == 1) && (br->GetColor(6, 9) == 3) )
        // II.b.1
        st.push_back(5);

    if ( (br->GetColor(5, 3) == 5) && (br->GetColor(2, 1) == 1) && (br->GetColor(6, 7) == 2) )
        // II.b.2
        st.push_back(6);

    if ( (br->GetColor(3, 3) == 3) && (br->GetColor(4, 1) == 1) && (br->GetColor(6, 3) == 4) )
        // II.b.3
        st.push_back(7);
}

```

```

if ( (br->GetColor(4, 3) == 4) && (br->GetColor(5, 1) == 1) && (br->GetColor(6, 1) == 5) )
// II.b.4
    st.push_back(8);

if ( (br->GetColor(2, 3) == 3) && (br->GetColor(3, 1) == 2) && (br->GetColor(6, 9) == 1) )
// II.c.1
    st.push_back(9);

if ( (br->GetColor(5, 3) == 2) && (br->GetColor(2, 1) == 5) && (br->GetColor(6, 7) == 1) )
// II.c.2
    st.push_back(10);

if ( (br->GetColor(3, 3) == 4) && (br->GetColor(4, 1) == 3) && (br->GetColor(6, 3) == 1) )
// II.c.3
    st.push_back(11);

if ( (br->GetColor(4, 3) == 5) && (br->GetColor(5, 1) == 4) && (br->GetColor(6, 1) == 1) )
// II.c.4
    st.push_back(12);

if (br->GetColor(2, 9) == 1)
    // II.d.1
    st.push_back(13);

if (br->GetColor(5, 9) == 1)
    // II.d.2
    st.push_back(14);

if (br->GetColor(3, 9) == 1)
    // II.d.3
    st.push_back(15);

if (br->GetColor(4, 9) == 1)
    // II.d.4
    st.push_back(16);

```

```

    if (br->GetColor(3, 7) == 1)
        // II.e.1
        st.push_back(17);

    if (br->GetColor(2, 7) == 1)
        // II.e.2
        st.push_back(18);

    if (br->GetColor(4, 7) == 1)
        // II.e.3
        st.push_back(19);

    if (br->GetColor(5, 7) == 1)
        // II.e.4
        st.push_back(20);

    if ( (br->GetColor(1, 1) == 1) && ((br->GetColor(5, 9) != 5) || (br->GetColor(2, 7) != 2))
)// II.f.1
        st.push_back(21);

    if ( (br->GetColor(1, 3) == 1) && ((br->GetColor(2, 9) != 2) || (br->GetColor(3, 7) != 3))
)// II.f.2
        st.push_back(22);

    if ( (br->GetColor(1, 9) == 1) && ((br->GetColor(3, 9) != 3) || (br->GetColor(4, 7) != 4))
)// II.f.3
        st.push_back(23);

    if ( (br->GetColor(1, 7) == 1) && ((br->GetColor(4, 9) != 4) || (br->GetColor(5, 7) != 5))
)// II.f.4
        st.push_back(24);

    if (st.empty())
        st.push_back(25);

```

// Если ни один из тех, что выше

```

        return st;
    }

list <int> private CAutoBrick::x3()
/*
    Метод для выяснения условий перехода
*/
{
    list <int> st;

    if ( (br->GetColor(2, 2) == 2) && (br->GetColor(6, 8) == 5) )           // III.a.1
        st.push_back(1);

    if ( (br->GetColor(3, 2) == 3) && (br->GetColor(6, 6) == 2) )           // III.a.2
        st.push_back(2);

    if ( (br->GetColor(4, 2) == 4) && (br->GetColor(6, 2) == 3) )           // III.a.3
        st.push_back(3);

    if ( (br->GetColor(5, 2) == 5) && (br->GetColor(6, 4) == 4) )           // III.a.4
        st.push_back(4);

    if ( (br->GetColor(2, 2) == 2) && (br->GetColor(6, 8) == 3) )           // III.b.1
        st.push_back(5);

    if ( (br->GetColor(5, 2) == 5) && (br->GetColor(6, 4) == 2) )           // III.b.2
        st.push_back(6);

    if ( (br->GetColor(4, 2) == 4) && (br->GetColor(6, 2) == 5) )           // III.b.3
        st.push_back(7);

    if ( (br->GetColor(3, 2) == 3) && (br->GetColor(6, 6) == 4) )           // III.b.4
        st.push_back(8);
}

```

```

if (st.empty())                                     // Если ни один из тех, что выше
    st.push_back(13);

if ( (br->GetColor(3, 4) != 3) || (br->GetColor(2, 6) != 2) )    // III.c.1
    st.push_back(9);

if ( (br->GetColor(2, 4) != 2) || (br->GetColor(5, 6) != 5) )    // III.c.2
    st.push_back(10);

if ( (br->GetColor(5, 4) != 5) || (br->GetColor(4, 6) != 4) )    // III.c.3
    st.push_back(11);

if ( (br->GetColor(4, 4) != 4) || (br->GetColor(3, 6) != 3) )    // III.c.4
    st.push_back(12);

return st;
}

list <int> private CAutoBrick::x4()
/*
    Метод для выяснения условий перехода
*/
{
    list <int> st;

    if ( (br->GetColor(2, 2) == 6) && (br->GetColor(3, 2) == 6) &&
        (br->GetColor(6, 2) == 6) && (br->GetColor(6, 4) == 6) )    // IV.a.1
        st.push_back(1);

    if ( (br->GetColor(2, 2) == 6) && (br->GetColor(5, 2) == 6) &&
        (br->GetColor(6, 2) == 6) && (br->GetColor(6, 6) == 6) )    // IV.a.2
        st.push_back(2);

    if ( (br->GetColor(4, 2) == 6) && (br->GetColor(3, 2) == 6) &&
        (br->GetColor(6, 4) == 6) && (br->GetColor(6, 8) == 6) )    // IV.a.3

```



```

        st.push_back(3);

    if ( (br->GetColor(4, 2) == 6) && (br->GetColor(5, 2) == 6) &&
        (br->GetColor(6, 6) == 6) && (br->GetColor(6, 8) == 6) ) // IV.a.4
        st.push_back(4);

    if ( (br->GetColor(2, 2) == 6) && (br->GetColor(4, 2) == 6) &&
        (br->GetColor(6, 4) == 6) && (br->GetColor(6, 6) == 6) ) // IV.b.1
        st.push_back(5);

    if ( (br->GetColor(3, 2) == 6) && (br->GetColor(5, 2) == 6) &&
        (br->GetColor(6, 2) == 6) && (br->GetColor(6, 8) == 6) ) // IV.b.2
        st.push_back(6);

    if ( (br->GetColor(2, 2) == 6) && (br->GetColor(3, 2) == 6) &&
        (br->GetColor(4, 2) == 6) && (br->GetColor(5, 2) == 6) ) // IV.c.1
        st.push_back(7);

    return st;
}

list <int> private CAutoBrick::x5()
/*
    Метод для выяснения условий перехода
*/
{
    list <int> st;

    if (br->GetColor(3, 2) == 2) // V.a.1
        st.push_back(1);

    if (br->GetColor(5, 2) == 2) // V.a.2
        st.push_back(2);

    if (br->GetColor(4, 2) == 2) // V.a.3
        st.push_back(3);
}

```

```

    if (br->GetColor(4, 2) == 3)                                // V.b.1
        st.push_back(4);

    if (br->GetColor(5, 2) == 3)                                // V.b.2
        st.push_back(5);

    if (br->GetColor(5, 2) == 4)                                // V.c.1
        st.push_back(6);

    return st;
}

list <int> private CAutoBrick::x6()
/*
Метод для выяснения условий перехода
*/
{
    list <int> st;

    if (IsEqual(br->GetColor(5, 3), br->GetColor(2, 1), br->GetColor(6, 7), 2, 3, 6))
        // VI.a.1
        st.push_back(1);

    if (IsEqual(br->GetColor(3, 3), br->GetColor(4, 1), br->GetColor(6, 3), 2, 3, 6))
        // VI.a.2
        st.push_back(2);

    if (IsEqual(br->GetColor(4, 3), br->GetColor(5, 1), br->GetColor(6, 1), 2, 3, 6))
        // VI.a.3
        st.push_back(3);

    if (IsEqual(br->GetColor(5, 3), br->GetColor(2, 1), br->GetColor(6, 7), 3, 4, 6))
        // VI.b.1
        st.push_back(4);
}

```

```

    if (IsEqual(br->GetColor(4, 3), br->GetColor(5, 1), br->GetColor(6, 1), 3, 4, 6))
        // VI.b.2
        st.push_back(5);

    return st;
}

list <int> private CAutoBrick::x7()
/*
    Метод для выяснения условий перехода
*/
{
    list <int> st;

    if (br->GetColor(3, 1) == 6) // VII.a.1
        st.push_back(1);

    if (br->GetColor(2, 3) == 6) // VII.a.2
        st.push_back(2);

    if ( (br->GetColor(2, 3) == 2) && (br->GetColor(2, 1) == 2) && // VII.b.1
        (br->GetColor(3, 1) == 3) && (br->GetColor(3, 3) == 3) &&
        (br->GetColor(4, 1) == 4) && (br->GetColor(4, 3) == 4) &&
        (br->GetColor(5, 1) == 5) && (br->GetColor(5, 3) == 5) )
        st.push_back(3);

    return st;
}

// Воздействия при переходе
list <action> private CAutoBrick::z1()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{

```

```

list <action> act;

switch (automata)
{
    case 0:
        AddAction(&act, 2, true);
        AddAction(&act, 2, true);
        break;
    case 1:
        AddAction(&act, 2, false);
        AddAction(&act, 6, false);
        AddAction(&act, 2, true);
        break;
    case 2:
        AddAction(&act, 6, false);
        AddAction(&act, 5, false);
        AddAction(&act, 6, true);
        AddAction(&act, 5, true);
        AddAction(&act, 6, true);
        AddAction(&act, 2, true);
        AddAction(&act, 6, false);
        AddAction(&act, 2, false);
        break;
    case 3:
        AddAction(&act, 2, true);
        AddAction(&act, 6, true);
        AddAction(&act, 3, true);
        AddAction(&act, 6, false);
        AddAction(&act, 3, false);
        AddAction(&act, 2, false);
        break;
    case 4:
        AddAction(&act, 6, true);
        AddAction(&act, 5, false);
        AddAction(&act, 6, true);
        AddAction(&act, 6, true);

```

```

        AddAction(&act, 5, true);
        AddAction(&act, 6, true);
        AddAction(&act, 5, false);
        AddAction(&act, 6, true);
        AddAction(&act, 5, true);
        break;
    case 5:
        AddAction(&act, 2, false);
        AddAction(&act, 5, false);
        AddAction(&act, 2, true);
        AddAction(&act, 3, false);
        AddAction(&act, 2, false);
        AddAction(&act, 5, true);
        AddAction(&act, 2, true);
        AddAction(&act, 3, true);
        break;
    case 6:
        AddAction(&act, 2, false);
        AddAction(&act, 3, true);
        AddAction(&act, 2, true);
        AddAction(&act, 3, false);
        AddAction(&act, 2, false);
        AddAction(&act, 3, true);
        AddAction(&act, 2, true);
        AddAction(&act, 3, false);
        break;
    default:
        break;
}

return act;
}

```

```

list <action> private CAutoBrick::z2()
/*

```

Повороты грани с номером ... по (true) или против (false) часовой стрелки

```

*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 3, true);
            AddAction(&act, 3, true);
            break;
        case 1:
            AddAction(&act, 5, false);
            AddAction(&act, 6, false);
            AddAction(&act, 5, true);
            break;
        case 2:
            AddAction(&act, 6, false);
            AddAction(&act, 2, false);
            AddAction(&act, 6, true);
            AddAction(&act, 2, true);
            AddAction(&act, 6, true);
            AddAction(&act, 3, true);
            AddAction(&act, 6, false);
            AddAction(&act, 3, false);
            break;
        case 3:
            AddAction(&act, 5, true);
            AddAction(&act, 6, true);
            AddAction(&act, 2, true);
            AddAction(&act, 6, false);
            AddAction(&act, 2, false);
            AddAction(&act, 5, false);
            break;
        case 4:
            AddAction(&act, 6, true);
            AddAction(&act, 4, false);
    }
}

```

```

        AddAction(&act, 6, true);
        AddAction(&act, 6, true);
        AddAction(&act, 4, true);
        AddAction(&act, 6, true);
        AddAction(&act, 4, false);
        AddAction(&act, 6, true);
        AddAction(&act, 4, true);
        break;
    case 5:
        AddAction(&act, 3, false);
        AddAction(&act, 2, false);
        AddAction(&act, 5, false);
        AddAction(&act, 2, true);
        AddAction(&act, 3, true);
        AddAction(&act, 2, false);
        AddAction(&act, 5, true);
        AddAction(&act, 2, true);
        break;
    case 6:
        AddAction(&act, 3, true);
        AddAction(&act, 2, false);
        AddAction(&act, 3, false);
        AddAction(&act, 2, true);
        AddAction(&act, 3, true);
        AddAction(&act, 2, false);
        AddAction(&act, 3, false);
        AddAction(&act, 2, true);
        break;
    default:
        break;
}

return act;
}

list <action> CAutoBrick::z3()

```

```

/*
Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 5, true);
            AddAction(&act, 5, true);
            break;
        case 1:
            AddAction(&act, 3, false);
            AddAction(&act, 6, false);
            AddAction(&act, 3, true);
            break;
        case 2:
            AddAction(&act, 6, false);
            AddAction(&act, 3, false);
            AddAction(&act, 6, true);
            AddAction(&act, 3, true);
            AddAction(&act, 6, true);
            AddAction(&act, 4, true);
            AddAction(&act, 6, false);
            AddAction(&act, 4, false);
            break;
        case 3:
            AddAction(&act, 3, true);
            AddAction(&act, 6, true);
            AddAction(&act, 4, true);
            AddAction(&act, 6, false);
            AddAction(&act, 4, false);
            AddAction(&act, 3, false);
            break;
        case 4:

```



```

        AddAction(&act, 6, true);
        AddAction(&act, 3, false);
        AddAction(&act, 6, true);
        AddAction(&act, 6, true);
        AddAction(&act, 3, true);
        AddAction(&act, 6, true);
        AddAction(&act, 3, false);
        AddAction(&act, 6, true);
        AddAction(&act, 3, true);

        AddAction(&act, 6, true);
        AddAction(&act, 4, false);
        AddAction(&act, 6, true);
        AddAction(&act, 6, true);
        AddAction(&act, 4, true);
        AddAction(&act, 6, true);
        AddAction(&act, 4, false);
        AddAction(&act, 6, true);
        AddAction(&act, 4, true);

        AddAction(&act, 6, true);
        AddAction(&act, 3, false);
        AddAction(&act, 6, true);
        AddAction(&act, 6, true);
        AddAction(&act, 3, true);
        AddAction(&act, 6, true);
        AddAction(&act, 3, false);
        AddAction(&act, 6, true);
        AddAction(&act, 3, true);
        break;
case 5:
        AddAction(&act, 2, false);
        AddAction(&act, 5, false);
        AddAction(&act, 4, false);
        AddAction(&act, 5, true);
        AddAction(&act, 2, true);

```

```

        AddAction(&act, 5, false);
        AddAction(&act, 4, true);
        AddAction(&act, 5, true);
        break;
    case 6:
        AddAction(&act, 6, false);
        break;
    default:
        break;
}

return act;
}

list <action> CAutoBrick::z4()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 4, true);
            AddAction(&act, 4, true);
            break;
        case 1:
            AddAction(&act, 4, false);
            AddAction(&act, 6, false);
            AddAction(&act, 4, true);
            break;
        case 2:
            AddAction(&act, 6, false);
            AddAction(&act, 4, false);
            AddAction(&act, 6, true);

```

```

        AddAction(&act, 4, true);
        AddAction(&act, 6, true);
        AddAction(&act, 5, true);
        AddAction(&act, 6, false);
        AddAction(&act, 5, false);
        break;
case 3:
        AddAction(&act, 4, true);
        AddAction(&act, 6, true);
        AddAction(&act, 5, true);
        AddAction(&act, 6, false);
        AddAction(&act, 5, false);
        AddAction(&act, 4, false);
        break;
case 4:
        AddAction(&act, 6, true);
        AddAction(&act, 2, false);
        AddAction(&act, 6, true);
        AddAction(&act, 6, true);
        AddAction(&act, 2, true);
        AddAction(&act, 6, true);
        AddAction(&act, 2, false);
        AddAction(&act, 6, true);
        AddAction(&act, 2, true);
        break;
case 5:
        AddAction(&act, 4, false);
        AddAction(&act, 3, false);
        AddAction(&act, 4, true);
        AddAction(&act, 5, false);
        AddAction(&act, 4, false);
        AddAction(&act, 3, true);
        AddAction(&act, 4, true);
        AddAction(&act, 5, true);
        break;
default:

```

```

        break;
    }

    return act;
}

list <action> CAutoBrick::z5()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 6, false);
            AddAction(&act, 3, false);
            AddAction(&act, 2, true);
            AddAction(&act, 3, true);
            break;
        case 1:
            AddAction(&act, 3, true);
            AddAction(&act, 6, true);
            AddAction(&act, 3, false);
            break;
        case 2:
            AddAction(&act, 6, true);
            AddAction(&act, 3, true);
            AddAction(&act, 6, false);
            AddAction(&act, 3, false);
            AddAction(&act, 6, false);
            AddAction(&act, 2, false);
            AddAction(&act, 6, true);
            AddAction(&act, 2, true);
            break;
    }
}

```

```

case 3:
    AddAction(&act, 2, true);
    AddAction(&act, 3, true);
    AddAction(&act, 6, true);
    AddAction(&act, 3, false);
    AddAction(&act, 6, false);
    AddAction(&act, 2, false);
    break;
case 4:
    AddAction(&act, 6, true);
    AddAction(&act, 4, false);
    AddAction(&act, 6, true);
    AddAction(&act, 6, true);
    AddAction(&act, 4, true);
    AddAction(&act, 6, true);
    AddAction(&act, 4, false);
    AddAction(&act, 6, true);
    AddAction(&act, 4, true);

    AddAction(&act, 6, true);
    AddAction(&act, 5, false);
    AddAction(&act, 6, true);
    AddAction(&act, 6, true);
    AddAction(&act, 5, true);
    AddAction(&act, 6, true);
    AddAction(&act, 5, false);
    AddAction(&act, 6, true);
    AddAction(&act, 5, true);

    AddAction(&act, 6, true);
    AddAction(&act, 4, false);
    AddAction(&act, 6, true);
    AddAction(&act, 6, true);
    AddAction(&act, 4, true);
    AddAction(&act, 6, true);
    AddAction(&act, 4, false);

```

```

        AddAction(&act, 6, true);
        AddAction(&act, 4, true);
        break;
    case 5:
        AddAction(&act, 5, false);
        AddAction(&act, 4, false);
        AddAction(&act, 3, false);
        AddAction(&act, 4, true);
        AddAction(&act, 5, true);
        AddAction(&act, 4, false);
        AddAction(&act, 3, true);
        AddAction(&act, 4, true);
        break;
    default:
        break;
}

return act;
}

list <action> CAutoBrick::z6()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 6, false);
            AddAction(&act, 4, false);
            AddAction(&act, 3, true);
            AddAction(&act, 4, true);
            break;
        case 1:

```

```

        AddAction(&act, 2, true);
        AddAction(&act, 6, true);
        AddAction(&act, 2, false);
        break;
case 2:
    AddAction(&act, 6, true);
    AddAction(&act, 2, true);
    AddAction(&act, 6, false);
    AddAction(&act, 2, false);
    AddAction(&act, 6, false);
    AddAction(&act, 5, false);
    AddAction(&act, 6, true);
    AddAction(&act, 5, true);
    break;
case 3:
    AddAction(&act, 3, true);
    AddAction(&act, 4, true);
    AddAction(&act, 6, true);
    AddAction(&act, 4, false);
    AddAction(&act, 6, false);
    AddAction(&act, 3, false);
    break;
case 4:
    AddAction(&act, 6, true);
    AddAction(&act, 3, false);
    AddAction(&act, 6, true);
    AddAction(&act, 6, true);
    AddAction(&act, 3, true);
    AddAction(&act, 6, true);
    AddAction(&act, 3, false);
    AddAction(&act, 6, true);
    AddAction(&act, 3, true);
    break;
default:
    break;
}

```

```

    return act;
}

list <action> private CAutoBrick::z7()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 6, false);
            AddAction(&act, 2, false);
            AddAction(&act, 5, true);
            AddAction(&act, 2, true);
            break;
        case 1:
            AddAction(&act, 4, true);
            AddAction(&act, 6, true);
            AddAction(&act, 4, false);
            break;
        case 2:
            AddAction(&act, 6, true);
            AddAction(&act, 5, true);
            AddAction(&act, 6, false);
            AddAction(&act, 5, false);
            AddAction(&act, 6, false);
            AddAction(&act, 4, false);
            AddAction(&act, 6, true);
            AddAction(&act, 4, true);
            break;
        case 3:
            AddAction(&act, 2, true);
    }
}

```



```

        AddAction(&act, 6, true);
        AddAction(&act, 3, true);
        AddAction(&act, 6, false);
        AddAction(&act, 3, false);
        AddAction(&act, 2, false);
        AddAction(&act, 3, true);
        AddAction(&act, 4, true);
        AddAction(&act, 6, true);
        AddAction(&act, 4, false);
        AddAction(&act, 6, false);
        AddAction(&act, 3, false);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z8()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 6, false);
            AddAction(&act, 5, false);
            AddAction(&act, 4, true);
            AddAction(&act, 5, true);
            break;
        case 1:
            AddAction(&act, 5, true);
    }
}

```

```

        AddAction(&act, 6, true);
        AddAction(&act, 5, false);
        break;
    case 2:
        AddAction(&act, 6, true);
        AddAction(&act, 4, true);
        AddAction(&act, 6, false);
        AddAction(&act, 4, false);
        AddAction(&act, 6, false);
        AddAction(&act, 3, false);
        AddAction(&act, 6, true);
        AddAction(&act, 3, true);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z9()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 2, false);
            AddAction(&act, 6, false);
            AddAction(&act, 2, true);
            break;
        case 1:
            AddAction(&act, 3, true);

```

```

        AddAction(&act, 6, false);
        AddAction(&act, 3, false);
        AddAction(&act, 6, true);
        AddAction(&act, 6, true);
        AddAction(&act, 3, true);
        AddAction(&act, 6, true);
        AddAction(&act, 3, false);
        break;
    case 2:
        AddAction(&act, 6, true);
        AddAction(&act, 3, true);
        AddAction(&act, 6, false);
        AddAction(&act, 3, false);
        AddAction(&act, 6, false);
        AddAction(&act, 2, false);
        AddAction(&act, 6, true);
        AddAction(&act, 2, true);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z10()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 3, false);

```

```

        AddAction(&act, 6, false);
        AddAction(&act, 3, true);
        break;
    case 1:
        AddAction(&act, 2, true);
        AddAction(&act, 6, false);
        AddAction(&act, 2, false);
        AddAction(&act, 6, true);
        AddAction(&act, 6, true);
        AddAction(&act, 2, true);
        AddAction(&act, 6, true);
        AddAction(&act, 2, false);
        break;
    case 2:
        AddAction(&act, 6, true);
        AddAction(&act, 2, true);
        AddAction(&act, 6, false);
        AddAction(&act, 2, false);
        AddAction(&act, 6, false);
        AddAction(&act, 5, false);
        AddAction(&act, 6, true);
        AddAction(&act, 5, true);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z11()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

```

```

switch (automata)
{
    case 0:
        AddAction(&act, 5, false);
        AddAction(&act, 6, false);
        AddAction(&act, 5, true);
        break;
    case 1:
        AddAction(&act, 4, true);
        AddAction(&act, 6, false);
        AddAction(&act, 4, false);
        AddAction(&act, 6, true);
        AddAction(&act, 6, true);
        AddAction(&act, 4, true);
        AddAction(&act, 6, true);
        AddAction(&act, 4, false);
        break;
    case 2:
        AddAction(&act, 6, true);
        AddAction(&act, 5, true);
        AddAction(&act, 6, false);
        AddAction(&act, 5, false);
        AddAction(&act, 6, false);
        AddAction(&act, 4, false);
        AddAction(&act, 6, true);
        AddAction(&act, 4, true);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z12()

```

```

/*
Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 4, false);
            AddAction(&act, 6, false);
            AddAction(&act, 4, true);
            break;
        case 1:
            AddAction(&act, 5, true);
            AddAction(&act, 6, false);
            AddAction(&act, 5, false);
            AddAction(&act, 6, true);
            AddAction(&act, 6, true);
            AddAction(&act, 5, true);
            AddAction(&act, 6, true);
            AddAction(&act, 5, false);
            break;
        case 2:
            AddAction(&act, 6, true);
            AddAction(&act, 4, true);
            AddAction(&act, 6, false);
            AddAction(&act, 4, false);
            AddAction(&act, 6, false);
            AddAction(&act, 3, false);
            AddAction(&act, 6, true);
            AddAction(&act, 3, true);
            break;
        default:
            break;
    }
}

```

```

        return act;
    }

list <action> private CAutoBrick::z13()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 2, true);
            AddAction(&act, 6, false);
            AddAction(&act, 2, false);
            break;
        case 1:
            AddAction(&act, 2, false);
            AddAction(&act, 6, false);
            AddAction(&act, 2, true);
            break;
        case 2:
            AddAction(&act, 6, true);
            break;
        default:
            break;
    }

    return act;
}

list <action> private CAutoBrick::z14()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/

```

```

*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 3, true);
            AddAction(&act, 6, false);
            AddAction(&act, 3, false);
            break;
        case 1:
            AddAction(&act, 5, false);
            AddAction(&act, 6, false);
            AddAction(&act, 5, true);
            break;
        default:
            break;
    }

    return act;
}

list <action> private CAutoBrick::z15()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 5, true);
            AddAction(&act, 6, false);
            AddAction(&act, 5, false);

```



```

        break;
    case 1:
        AddAction(&act, 3, false);
        AddAction(&act, 6, false);
        AddAction(&act, 3, true);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z16()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 4, true);
            AddAction(&act, 6, false);
            AddAction(&act, 4, false);
            break;
        case 1:
            AddAction(&act, 4, false);
            AddAction(&act, 6, false);
            AddAction(&act, 4, true);
            break;
        default:
            break;
    }
}

```

```

    return act;
}

list <action> private CAutoBrick::z17()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 2, true);
            AddAction(&act, 2, true);
            AddAction(&act, 6, false);
            AddAction(&act, 2, true);
            AddAction(&act, 2, true);
            break;
        case 1:
            AddAction(&act, 2, false);
            AddAction(&act, 6, false);
            AddAction(&act, 2, true);
            break;
        default:
            break;
    }

    return act;
}

list <action> private CAutoBrick::z18()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{

```

```

list <action> act;

switch (automata)
{
    case 0:
        AddAction(&act, 3, true);
        AddAction(&act, 3, true);
        AddAction(&act, 6, false);
        AddAction(&act, 3, true);
        AddAction(&act, 3, true);
        break;
    case 1:
        AddAction(&act, 5, false);
        AddAction(&act, 6, false);
        AddAction(&act, 5, true);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z19()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 5, true);
            AddAction(&act, 5, true);
            AddAction(&act, 6, false);

```

```

        AddAction(&act, 5, true);
        AddAction(&act, 5, true);
        break;
    case 1:
        AddAction(&act, 3, false);
        AddAction(&act, 6, false);
        AddAction(&act, 3, true);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z20()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 4, true);
            AddAction(&act, 4, true);
            AddAction(&act, 6, false);
            AddAction(&act, 4, true);
            AddAction(&act, 4, true);
            break;
        case 1:
            AddAction(&act, 4, false);
            AddAction(&act, 6, false);
            AddAction(&act, 4, true);
            break;
    }
}

```

```

        default:
            break;
    }

    return act;
}

list <action> private CAutoBrick::z21()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 2, true);
            AddAction(&act, 2, true);
            AddAction(&act, 6, false);
            AddAction(&act, 2, true);
            AddAction(&act, 2, true);
            break;
        case 1:
            AddAction(&act, 2, true);
            AddAction(&act, 6, false);
            AddAction(&act, 2, false);
            break;
        default:
            break;
    }

    return act;
}

list <action> private CAutoBrick::z22()

```

```

/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 3, true);
            AddAction(&act, 3, true);
            AddAction(&act, 6, false);
            AddAction(&act, 3, true);
            AddAction(&act, 3, true);
            break;
        case 1:
            AddAction(&act, 3, true);
            AddAction(&act, 6, false);
            AddAction(&act, 3, false);
            break;
        default:
            break;
    }

    return act;
}

list <action> private CAutoBrick::z23()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {

```

```

        case 0:
            AddAction(&act, 5, true);
            AddAction(&act, 5, true);
            AddAction(&act, 6, false);
            AddAction(&act, 5, true);
            AddAction(&act, 5, true);
            break;
        case 1:
            AddAction(&act, 4, true);
            AddAction(&act, 6, false);
            AddAction(&act, 4, false);
            break;
        default:
            break;
    }

    return act;
}

list <action> private CAutoBrick::z24()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 4, true);
            AddAction(&act, 4, true);
            AddAction(&act, 6, false);
            AddAction(&act, 4, true);
            AddAction(&act, 4, true);
            break;
        case 1:

```

```

        AddAction(&act, 5, true);
        AddAction(&act, 6, false);
        AddAction(&act, 5, false);
        break;
    default:
        break;
}

return act;
}

list <action> private CAutoBrick::z25()
/*
    Повороты грани с номером ... по (true) или против (false) часовой стрелки
*/
{
    list <action> act;

    switch (automata)
    {
        case 0:
            AddAction(&act, 6, true);
            break;
        case 1:
            AddAction(&act, 6, true);
            break;
        default:
            break;
    }

    return act;
}

list <action> private CAutoBrick::z(int i)
/*
    Поворот грани с номером ... по (true) или против (false) часовой стрелки
*/

```



```
*/
{
    switch (i)
    {
        case 1:
            return z1();
        case 2:
            return z2();
        case 3:
            return z3();
        case 4:
            return z4();
        case 5:
            return z5();
        case 6:
            return z6();
        case 7:
            return z7();
        case 8:
            return z8();
        case 9:
            return z9();
        case 10:
            return z10();
        case 11:
            return z11();
        case 12:
            return z12();
        case 13:
            return z13();
        case 14:
            return z14();
        case 15:
            return z15();
        case 16:
            return z16();
    }
}
```

```

        case 17:
            return z17();
        case 18:
            return z18();
        case 19:
            return z19();
        case 20:
            return z20();
        case 21:
            return z21();
        case 22:
            return z22();
        case 23:
            return z23();
        case 24:
            return z24();
        case 25:
            return z25();
    }
}

```

```

// Вспомогательные функции
void CAutoBrick::AddAction(list <action>* act, int side, bool direct)
/*
    Добавляет в список действий новый поворот
*/
{
    action rot;
    rot.side    = side;
    rot.direct  = direct;

    act->push_back(rot);
}

bool CAutoBrick::IsIn(int i, list <int> st)

```

```

/*
    Проверяет находится ли i в листе st
*/
{
    for (list<int> :: iterator j = st.begin(); j != st.end(); j++)
    {
        if (*j == i)
        {
            return true;
        }
    }

    return false;
}

bool CAutoBrick::IsEqual(int x1, int x2, int x3, int y1, int y2, int y3)
/*
    Проверяет на совпадение две тройки
*/
{
    if ( ( (x1 == y1) && (x2 == y2) && (x3 == y3) ) ||
        ( (x1 == y1) && (x2 == y3) && (x3 == y2) ) ||
        ( (x1 == y2) && (x2 == y1) && (x3 == y3) ) ||
        ( (x1 == y2) && (x2 == y3) && (x3 == y1) ) ||
        ( (x1 == y3) && (x2 == y1) && (x3 == y2) ) ||
        ( (x1 == y3) && (x2 == y1) && (x3 == y2) ) )
        return true;

    return false;
}

// BigBrick.h: описание класса CBigBrick
//
////////////////////////////////////

```

```

#if !defined(AFX_BIGBRICK_H__D0A02268_1759_4DD7_AE50_8C941C45D1C1__INCLUDED_)
#define AFX_BIGBRICK_H__D0A02268_1759_4DD7_AE50_8C941C45D1C1__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "SmallBrick.h"
#include "Col.h"

class CBigBrick
{
public:
    CBigBrick();
    virtual ~CBigBrick();
// Поля
private:
    CSmallBrick bricks[3][3][3];
    float nSize;
    float nX;
    float nY;
    float nZ;
    float nDelta;

// Методы
private:
    void Refresh();
    // повороты граней
    void RotateXNeg(int side);
    void RotateXPos(int side);
    void RotateYNeg(int side);
    void RotateYPos(int side);
    void RotateZNeg(int side);
    void RotateZPos(int side);

public:

```

```

void SetX(float x);
void SetY(float y);
void SetZ(float z);
void SetSize(float size);
void SetDelta(float d);
float GetX();
float GetY();
float GetZ();
float GetSize();
float GetDelta();
void Draw();
void Init(); // инициализация всех элементов класса

void Rotate(int side, bool dir);
int GetColor(int x, int y, int z, int side); // возвращает цвет грани side с координатами
(x,y,z)
int GetColor(int side, int num); // Возвращает цвет грани кубика с номером num
из грани side
void SetColor(int x, int y, int z, int side, int col);
};

#endif // !defined(AFX_BIGBRICK_H__D0A02268_1759_4DD7_AE50_8C941C45D1C1__INCLUDED_)

// BigBrick.cpp: реализация класса CBigBrick
//
////////////////////////////////////

#include "stdafx.h"
#include "BigBrick.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW

```

```

#endif

CBigBrick::CBigBrick()
{
}

CBigBrick::~~CBigBrick()
{
}

void CBigBrick::SetX(float x)
/*
    Устанавливаем x - координату
*/
{
    nX = x;
}

void CBigBrick::SetY(float y)
/*
    Устанавливаем y - координату
*/
{
    nY = y;
}

void CBigBrick::SetZ(float z)
/*
    Устанавливаем z - координату
*/
{
    nZ = z;
}

void CBigBrick::SetSize(float size)

```

```

/*
    Устанавливает размер грани куба
*/
{
    nSize = size;
}

void CBigBrick::SetDelta(float d)
{
    nDelta = d;
}

float CBigBrick::GetX()
/*
    Возвращает x - координату
*/
{
    return nX;
}

float CBigBrick::GetY()
/*
    Возвращает y - координату
*/
{
    return nY;
}

float CBigBrick::GetZ()
/*
    Возвращает z - координату
*/
{
    return nZ;
}

```

```

float CBigBrick::GetSize()
/*
    Возвращает размер куба
*/
{
    return nSize;
}

float CBigBrick::GetDelta()
{
    return nDelta;
}


void CBigBrick::Init()
/*
    Первоначальная инициализация куба
*/
{
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            for(int k = 0; k < 3; k++)
            {
                bricks[i][j][k].SetSize(nSize);
                bricks[i][j][k].SetX(i);
                bricks[i][j][k].SetY(j);
                bricks[i][j][k].SetZ(k);
                bricks[i][j][k].SetColor();
                bricks[i][j][k].Refresh();
            }
        }
    }
}

```



```

    }
}

void CBigBrick::Draw()
/*
    Метод отрисовывает куб
*/
{
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            for(int k = 0; k < 3; k++)
            {
                bricks[i][j][k].Draw();
            }
        }
    }
}

int CBigBrick::GetColor(int x, int y, int z, int side)
/*
    Возвращает цвет в виде структуры color грани кубика
    с номером side и координатами (x,y,z)
*/
{
    return bricks[x][y][z].GetColor(side);
}

void CBigBrick::SetColor(int x, int y, int z, int side, int col)
{
    bricks[x][y][z].SetColor(side, col);
}

int CBigBrick::GetColor(int side, int num)

```

```

/*
    Возвращает цвет грани маленького кубика с номером num на грани side
*/
{
    int t_x;
    int t_y;
    int tx;
    int ty;
    int tz;

    t_x = (num - 1) - 3 * ((num - 1) / 3);
    t_y = 2 - (num - 1) / 3;

    switch(side - 1)
    {
        case 0:
            tx = t_x;
            ty = 0;
            tz = t_y;
            break;
        case 1:
            tx = t_x;
            ty = t_y;
            tz = 2;
            break;
        case 2:
            tx = 2;
            ty = t_y;
            tz = 2 - t_x;
            break;
        case 3:
            tx = 2 - t_x;
            ty = t_y;
            tz = 0;
            break;
        case 4:

```

```

        tx = 0;
        ty = t_y;
        tz = t_x;
        break;
    case 5:
        tx = t_x;
        ty = 2;
        tz = 2 - t_y;
        break;
}

return (bricks[tx][ty][tz].GetColor(side - 1) + 1);
}

```

```

// Повороты
void CBigBrick::Refresh()
/*
    Перестроение массива bricks
*/
{
    int x;
    int y;
    int z;
    CSmallBrick tempBr[3][3][3];

    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            for(int k = 0; k < 3; k++)
            {
                x = bricks[i][j][k].GetX();
                y = bricks[i][j][k].GetY();
                z = bricks[i][j][k].GetZ();
                tempBr[x][y][z] = bricks[i][j][k];
            }
        }
    }
}

```

```

        }
    }
}

for(int i = 0; i < 3; i++)
{
    for(int j = 0; j < 3; j++)
    {
        for(int k = 0; k < 3; k++)
        {
            bricks[i][j][k] = tempBr[i][j][k];
        }
    }
}

}

void CBigBrick::RotateXNeg(int side)
/*
    Поворот кубика вокруг оси у в отрицательном направлении
*/
{
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            bricks[side][i][j].RotateXNeg();
        }
    }
    Refresh();
}

void CBigBrick::RotateXPos(int side)
/*
    Поворот кубика вокруг оси у в положительном направлении
*/

```

```

{
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            bricks[side][i][j].RotateXPos();
        }
    }
    Refresh();
}

void CBigBrick::RotateYNeg(int side)
/*
    Поворот кубика вокруг оси y в отрицательном направлении
*/
{
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            bricks[i][side][j].RotateYNeg();
        }
    }
    Refresh();
}

void CBigBrick::RotateYPos(int side)
/*
    Поворот кубика вокруг оси y в положительном направлении
*/
{
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {

```

```

        bricks[i][side][j].RotateYPos();
    }
}
Refresh();
}

void CBigBrick::RotateZNeg(int side)
/*
    Поворот кубика вокруг оси z в отрицательном направлении
*/
{
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            bricks[j][i][side].RotateZNeg();
        }
    }
    Refresh();
}

void CBigBrick::RotateZPos(int side)
/*
    Поворот кубика вокруг оси z в положительном направлении
*/
{
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            bricks[j][i][side].RotateZPos();
        }
    }
    Refresh();
}

```

```
void CBigBrick::Rotate(int side, bool dir)
```

```
/*  
    Метод занимается поворотом грани с номером side.  
    Если dir, то в положительном направлении, в ином случае  
    в отрицательном. Выбор ориентации поворота выбирается исходя из того, что  
    мы смотрим на грань
```

```
*/
```

```
{
```

```
    switch(side - 1)
```

```
    {
```

```
    case 0:
```

```
    {
```

```
        if (!dir) RotateYPos(0);
```

```
        else RotateYNeg(0);
```

```
        break;
```

```
    }
```

```
    case 1:
```

```
    {
```

```
        if (!dir) RotateZPos(2);
```

```
        else RotateZNeg(2);
```

```
        break;
```

```
    }
```

```
    case 2:
```

```
    {
```

```
        if (dir) RotateXPos(2);
```

```
        else RotateXNeg(2);
```

```
        break;
```

```
    }
```

```
    case 3:
```

```
    {
```

```
        if (dir) RotateZPos(0);
```

```
        else RotateZNeg(0);
```

```
        break;
```

```
    }
```

```
    case 4:
```

```

        {
            if (!dir) RotateXPos(0);
            else RotateXNeg(0);
            break;
        }
case 5:
    {
        if (dir) RotateYPos(2);
        else RotateYNeg(2);
        break;
    }
}
}

```

```

// SmallBrick.h: описание класса CSmallBrick
//

```

```

////////////////////////////////////

```

```

#if !defined(AFX_SMALLBRICK_H__A7B6E5B4_6A6B_4124_858B_1ACAA43B4500__INCLUDED_)
#define AFX_SMALLBRICK_H__A7B6E5B4_6A6B_4124_858B_1ACAA43B4500__INCLUDED_

```

```

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include <gl\gl.h>
#include <gl\glu.h>
#include <gl\glaux.h>
#include "Col.h"

```

```

// Заголовочный файл для OpenGL32 библиотеки
// Заголовочный файл для GLu32 библиотеки
// Заголовочный файл для GLaux библиотеки

```

```

struct color
{
    float red;
    float green;
    float blue;
};

```



```

class CSmallBrick
{
public:
    CSmallBrick();
    virtual ~CSmallBrick();

// Поля
private:
    CCol nColor[6];

    float nSize;
    float nX;      // x - координата
    float nY;      // y - координата
    float nZ;      // z - координата
    float nDelta;
    int x;
    int y;
    int z;

// Методы
private:
    // Пересчитывает координаты кубика при поворотах
    void RotateNeg(int &t, int &r);           // в отрицательном направлении
    void RotatePos(int &t, int &r);          // в положительном направлении

public:
    void SetX(int tx);                       // устанавливает координаты
    void SetY(int ty);
    void SetZ(int tz);
    void SetSize(float size);                // устанавливает размер
    void SetDelta(float d);
    void SetColor();                         // устанавливает цвет граней кубика
    void SetColor(int side, int color);
    int GetX();                             // возвращает координаты
    int GetY();

```

```

    int GetZ();
    float GetSize(); // возвращает размер
    float GetDelta();
    void Draw(); // отрисовывает кубик
    void Refresh(); // пересчитывает координаты кубика
    int GetColor(int side); // возвращает цвет грани с номером side

    // Повороты
    void RotateXPos(); // вокруг оси x в положительном направлении
    void RotateXNeg(); // вокруг оси x в отрицательном направлении
    void RotateYPos(); // вокруг оси y в положительном направлении
    void RotateYNeg(); // вокруг оси y в отрицательном направлении
    void RotateZPos(); // вокруг оси z в положительном направлении
    void RotateZNeg(); // вокруг оси z в отрицательном направлении

};

#endif // !defined(AFX_SMALLBRICK_H__A7B6E5B4_6A6B_4124_858B_1ACAA43B4500__INCLUDED_)

// SmallBrick.cpp: реализация класса CSmallBrick
//
////////////////////////////////////

#include "stdafx.h"
#include "SmallBrick.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

CSmallBrick::CSmallBrick()
{

```

```

        x = 0;
        y = 0;
        z = 0;
        nSize = 0;
    }

CSmallBrick::~CSmallBrick()
{
}

void CSmallBrick::SetX(int tx)
/*
    Устанавливаем x - координату
*/
{
    x = tx;
}

void CSmallBrick::SetY(int ty)
/*
    Устанавливаем y - координату
*/
{
    y = ty;
}

void CSmallBrick::SetZ(int tz)
/*
    Устанавливаем z - координату
*/
{
    z = tz;
}

void CSmallBrick::SetSize(float size)

```

```

/*
    Устанавливает размер грани куба
*/
{
    nSize = size;
}

void CSmallBrick::SetDelta(float d)
{
    nDelta = d;
}

void CSmallBrick::SetColor()
{
    for(int i = 0; i < 6; i++)
    {
        nColor[i].SetNum(i);
    }
}

void CSmallBrick::SetColor(int side, int color)
{
    nColor[side].SetNum(color);
}

int CSmallBrick::GetX()
/*
    Возвращает x - координату
*/
{
    return x;
}

int CSmallBrick::GetY()
/*

```

```

        Возвращает y - координату
*/
{
    return y;
}

int CSmallBrick::GetZ()
/*
    Возвращает z - координату
*/
{
    return z;
}

float CSmallBrick::GetSize()
/*
    Возвращает размер куба
*/
{
    return nSize;
}

float CSmallBrick::GetDelta()
{
    return nDelta;
}

int CSmallBrick::GetColor(int side)
/*
    Возвращает цвет грани кубика с номером side
*/
{
    return nColor[side].GetNum();
}

```

```

void CSmallBrick::Refresh()
{
    nX = (x-1.5f) * nSize;
    nY = (y-1.5f) * nSize;
    nZ = (z-1.5f) * nSize;
}

void CSmallBrick::Draw()
/*
    Метод отрисовывает куб
*/
{
    // передняя грань
    glColor3f(nColor[1].red, nColor[1].green, nColor[1].blue);
    glBegin(GL_QUADS);
        glVertex3f(nX, nY, nZ);
        glVertex3f(nX, nY + nSize, nZ);
        glVertex3f(nX + nSize, nY + nSize, nZ);
        glVertex3f(nX + nSize, nY, nZ);
    glEnd();

    // левая грань
    glColor3f(nColor[4].red, nColor[4].green, nColor[4].blue);
    glBegin(GL_QUADS);
        glVertex3f(nX, nY, nZ);
        glVertex3f(nX, nY, nZ - nSize);
        glVertex3f(nX, nY + nSize, nZ - nSize);
        glVertex3f(nX, nY + nSize, nZ);
    glEnd();

    // правая грань
    glColor3f(nColor[2].red, nColor[2].green, nColor[2].blue);
    glBegin(GL_QUADS);
        glVertex3f(nX + nSize, nY, nZ);
        glVertex3f(nX + nSize, nY, nZ - nSize);

```

```

        glVertex3f(nX + nSize, nY + nSize, nZ - nSize);
        glVertex3f(nX + nSize, nY + nSize, nZ);
    glEnd();

    // задняя грань
    glColor3f(nColor[3].red, nColor[3].green, nColor[3].blue);
    glBegin(GL_QUADS);
        glVertex3f(nX, nY, nZ - nSize);
        glVertex3f(nX, nY + nSize, nZ - nSize);
        glVertex3f(nX + nSize, nY + nSize, nZ - nSize);
        glVertex3f(nX + nSize, nY, nZ - nSize);
    glEnd();

    // верхняя грань
    glColor3f(nColor[5].red, nColor[5].green, nColor[5].blue);
    glBegin(GL_QUADS);
        glVertex3f(nX, nY + nSize, nZ);
        glVertex3f(nX, nY + nSize, nZ - nSize);
        glVertex3f(nX + nSize, nY + nSize, nZ - nSize);
        glVertex3f(nX + nSize, nY + nSize, nZ);
    glEnd();

    // нижняя грань
    glColor3f(nColor[0].red, nColor[0].green, nColor[0].blue);
    glBegin(GL_QUADS);
        glVertex3f(nX, nY, nZ);
        glVertex3f(nX, nY, nZ - nSize);
        glVertex3f(nX + nSize, nY, nZ - nSize);
        glVertex3f(nX + nSize, nY, nZ);
    glEnd();
}

// ПОВОРОТЫ
void CSmallBrick::RotateNeg(int &t, int&r)
/*

```

Изменение координат при повороте в отрицательном направлении

```
*/
{
    if((t == 0)&&( r== 0)) { t = 2; return;}
    if((t == 1)&&( r== 0)) { t = 2; r = 1; return;}
    if((t == 2)&&( r== 0)) { r = 2; return;}
    if((t == 2)&&( r== 1)) { t = 1; r = 2; return;}
    if((t == 2)&&( r== 2)) { t = 0; return;}
    if((t == 1)&&( r== 2)) { t = 0; r = 1; return;}
    if((t == 0)&&( r== 2)) { t = 0; r = 0; return;}
    if((t == 0)&&( r== 1)) { t = 1; r = 0; return;}

}
```

void CSmallBrick::RotatePos(int &t, int&r)

/\*  
Изменение координат при повороте в положительном направлении

```
*/
{
    if((t == 0)&&( r== 0)) { r = 2; return;}
    if((t == 1)&&( r== 0)) { t = 0; r = 1; return;}
    if((t == 2)&&( r== 0)) { t = 0; return;}
    if((t == 2)&&( r== 1)) { t = 1; r = 0; return;}
    if((t == 2)&&( r== 2)) { r = 0; return;}
    if((t == 1)&&( r== 2)) { t = 2; r = 1; return;}
    if((t == 0)&&( r== 2)) { t = 2; return;}
    if((t == 0)&&( r== 1)) { t = 1; r = 2; return;}

}
```

void CSmallBrick::RotateXNeg()

/\*  
Поворот кубика по оси x в отрицательную сторону  
\*/  
{



```

    CCol tCol;
    tCol = nColor[0];
    nColor[0] = nColor[1];
    nColor[1] = nColor[5];
    nColor[5] = nColor[3];
    nColor[3] = tCol;
    RotateNeg(y, z);
    Refresh();
}

void CSmallBrick::RotateXPos()
/*
    Поворот кубика по оси x в положительном направлении
*/
{
    CCol tCol;
    tCol = nColor[0];
    nColor[0] = nColor[3];
    nColor[3] = nColor[5];
    nColor[5] = nColor[1];
    nColor[1] = tCol;
    RotatePos(y, z);
    Refresh();
}

void CSmallBrick::RotateYNeg()
/*
    Поворот кубика по оси y в отрицательную сторону
*/
{
    CCol tCol;
    tCol = nColor[1];
    nColor[1] = nColor[4];
    nColor[4] = nColor[3];
    nColor[3] = nColor[2];
    nColor[2] = tCol;

```

```

        RotateNeg(z, x);
        Refresh();
    }

void CSmallBrick::RotateYPos()
/*
    Поворот кубика по оси y в положительном направлении
*/
{
    CCol tCol;
    tCol = nColor[1];
    nColor[1] = nColor[2];
    nColor[2] = nColor[3];
    nColor[3] = nColor[4];
    nColor[4] = tCol;
    RotatePos(z, x);
    Refresh();
}

void CSmallBrick::RotateZNeg()
/*
    Поворот кубика по оси z в отрицательную сторону
*/
{
    CCol tCol;
    tCol = nColor[0];
    nColor[0] = nColor[2];
    nColor[2] = nColor[5];
    nColor[5] = nColor[4];
    nColor[4] = tCol;
    RotateNeg(y, x);
    Refresh();
}

void CSmallBrick::RotateZPos()
/*

```

```

        Поворот кубика по оси z в положительном направлении
*/
{
    CCol tCol;
    tCol = nColor[0];
    nColor[0] = nColor[4];
    nColor[4] = nColor[5];
    nColor[5] = nColor[2];
    nColor[2] = tCol;
    RotatePos(y, x);
    Refresh();
}

// Col.h: описание класса CCol
//
////////////////////////////////////

#pragma once

class CCol
{
public:
    CCol(void);
    ~CCol(void);

// Поля
private:
    int nNum;                // номер цвета
public:
    float red;
    float green;
    float blue;

// Методы
public:

```

```

        void SetNum(int n);
        int GetNum();
};

// Col.cpp: реализация класса CCol
//
////////////////////////////////////

#include "StdAfx.h"
#include "col.h"

CCol::CCol(void)
{
}

CCol::~~CCol(void)
{
}

int CCol::GetNum()
/*
    Получить цвет граней
*/
{
    return nNum;
}

void CCol::SetNum(int n)
/*
    Установить цвет граней
*/
{
    nNum = n;
    switch (n)
    {

```

```
case 0:
{
    red    = 1.0f;
    blue   = 1.0f;
    green  = 1.0f;
    break;
}
case 1:
{
    red    = 0.0f;
    blue   = 1.0f;
    green  = 0.0f;
    break;
}
case 2:
{
    red    = 200.0 / 255.0;
    blue   = 150.0 / 255.0;
    green  = 20.0  / 255.0;
    break;
}
case 3:
{
    red    = 0.0f;
    blue   = 0.0f;
    green  = 1.0f;
    break;
}
case 4:
{
    red    = 1.0f;
    blue   = 0.0f;
    green  = 0.0f;
    break;
}
case 5:
```

```
    {  
        red    = 1.0f;  
        blue   = 0.0f;  
        green  = 1.0f;  
        break;  
    }  
}
```