**Задание:** используя API ОС Mac OS, создать графическую программу, которая создает пустое окно, в котором можно по нажатию клавиш F1, F2, F3 создать различные фигуры: треугольник, квадрат и ромб. Также требуется добавить возможность выделения активной фигуры красной заливкой и тонкой черной обводкой, а неактивная фигура должна представлять из себя только обводку без внутренней заливки. Следует предусмотреть перемещение активной фигуры по нажатию ЛКМ, отражение фигуры относительно точки клика по нажатию ПКМ, возможность увеличения и уменьшения активной фигуры с помощью стрелок, а также выбора активной фигуры с помощью других стрелок.

**Содержание**

[1. Геометрическая постановка задачи 4](#_Toc133243265)

[2.1 Описание переменных или информационной модели 5](#_Toc133243266)

[2.2 Описание функций программы 5](#_Toc133243267)

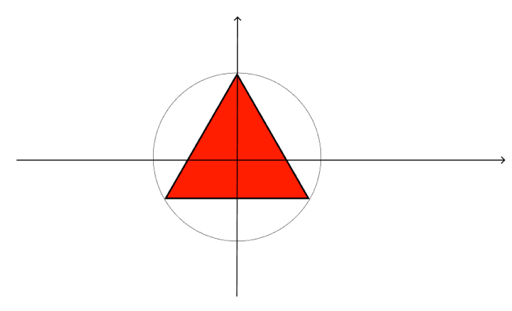
[2.3 Описание основных средств API 6](#_Toc133243268)

[3. Описание порядка работы пользователя 7](#_Toc133243269)

[4. Блок схемы алгоритма 8](#_Toc133243270)

[5. Список литературы 10](#_Toc133243271)

# Геометрическая постановка задачи

Построение фигур в программе осуществляется с помощью координат и окружности. Благодаря геометрическим параметрам, заданным в коде, и геометрическим расчетам координаты для фигур представлены следующим образом:

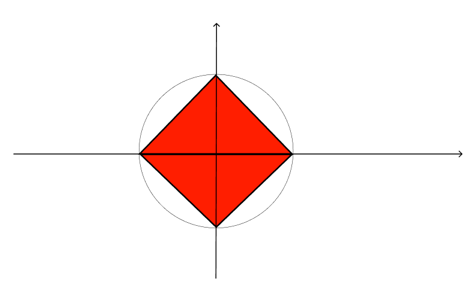
1. Для треугольника:

x[i] - R[i] \* , y[i] + R[i] \*

x[i] + R[i] \* , y[i] + R[i] \*

x[i], y[i] - R[i])

Рисунок 1 – график рисования

 треугольника

1. Для ромба:

x[i], y[i] - R[i])

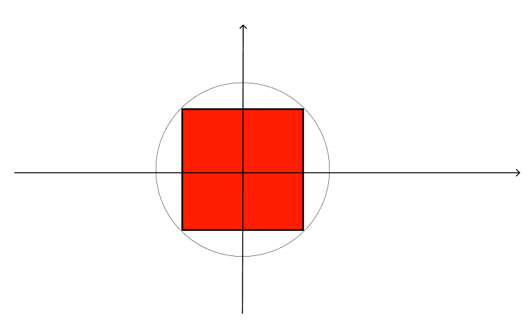
x[i] + R[i], y[i])

x[i], y[i] + R[i])

x[i] - R[i], y[i])

Рисунок 2 – график рисования

ромба



1. Для квадрата:

x[i] - R[i] \* , y[i] + R[i] \*

x[i] + R[i] \* , y[i] + R[i] \*

x[i] + R[i] \* , y[i] - R[i] \*

x[i] - R[i] \* , y[i] - R[i] \*

Рисунок 3 – график рисования

квадрата

Перемещение фигур осуществляется с помощью поиска точки клика ЛКМ и перемещения центра фигуры в эту точку. Активная фигура выделяется путем присвоения переменной, отвечающей за активность фигур, значения 1 активной фигуре, и значения 0 всем неактивным фигурам.

**2. Структура программы**

# 2.1 Описание переменных или информационной модели

Код данной программы содержит в себе массивы целых чисел V[N\_MAX], x[N\_MAX], y[N\_MAX], R[N\_MAX], t[N\_MAX], a[N\_MAX], которые предназначены для обозначения видимости, точек фигур, размера и т.д., размер массива N\_MAX = 10, для ограничения по созданию фигур. Также в коде задействована переменная m, которая является целым числом, представляющим индекс текущего активного треугольника в массивах. Переменная n используется для обозначения общего количества созданных фигур. Переменная f выполняет роль переключателя между режимами отражения фигуры в осях. Переменная p используется для обозначения числа ПИ.

Данный код представляет простую графическую программу на языке Objective-C, использующую Cocoa фреймворк для создания и отображения фигур на экране.

Программа имеет возможность создавать треугольники, квадраты и ромбы на экране, изменять их размер и положение с помощью клавиш и мыши, и менять направление отражения треугольников с помощью f.

Все параметры треугольников, такие как координаты, радиус, ширина, активность хранятся в соответствующих массивах, и обновляются после выполнения действий пользователя

# 2.2 Описание функций программы

Этот код Objective-C определяет пользовательский класс представления под названием TriangleView, который наследуется от NSView и реализует несколько функций:

- initWithFrame:: это специальный "запускающий" метод, который вызывается при создании нового объекта класса. Он инициализирует некоторые переменные, необходимые для работы объекта, и возвращает готовый к использованию объект.

- acceptsFirstResponder: Этот метод возвращает логическое значение, указывающее, может ли объект стать первым, кто будет обрабатывать события нажатия клавиш на клавиатуре. (Если его не добавить в код, то keyDown не будет работать, а следовательно действия по нажатию клавиш выполняться не будут).

- keyDown:: Этот метод вызывается при нажатии клавиши. Он проверяет, какая клавиша была нажата, и выполняет различные действия на основе кода клавиши, такие как добавление новой фигуры, перемещение текущей фигуры или удаление фигуры.

- mouseDown:: Этот метод вызывается при нажатии левой кнопки мыши. Он записывает положение клика мыши на окне и использует его для обновления положения текущей фигуры.

- rightMouseDown:: Этот метод вызывается при нажатии правой кнопки мыши. Он записывает положение мыши и использует его для обновления положения и отражения текущей фигуры.

- drawRect:: Этот метод вызывается, когда требуется перерисовать содержимое окна. Он перебирает все добавленные фигуры и рисует их с помощью NSBezierPath. Конкретная фигура, которая будет нарисована, определяется "типом" фигуры (хранится в массиве t), а также размером и положением фигуры (хранятся в других массивах, указанных ранее).

Также код содержит реализацию следующих функций программы:

- applicationDidFinishLaunching: Эта функция вызывается после того, как приложение успешно загрузилось и готово к работе. Внутри данной функции создается окно с определенными настройками (размер, стиль и т. д.), устанавливается заголовок окна, создается и добавляется на окно экземпляр класса TriangleView (представление фигур), и окно делается видимым и активным.

- applicationWillTerminate: Эта функция вызывается перед закрытием приложения. В данной реализации она оставлена пустой, но в ней можно разместить код для корректного завершения работы приложения, освобождения ресурсов, сохранения данных и т. д.

# 2.3 Описание основных средств API

Код написан на языке Objective-C для создания пользовательского интерфейса Mac OS. Используется фреймворк Cocoa для создания и управления графическими элементами интерфейса.

Основные средства API, используемые в этом коде:

NSRect - структура, которая представляет прямоугольную область в системе координат Cocoa.

NSObject - базовый класс для всех объектов Cocoa.

NSView - класс для создания и управления графическими элементами интерфейса.

NSEvent - класс для представления событий ввода, таких как нажатия клавиш и движения мыши.

NSBezierPath - класс для создания и редактирования путей в двумерном пространстве, которые могут быть использованы для рисования графических элементов.

NSColor - класс для представления цвета в Cocoa.

# 3. Описание порядка работы пользователя

После запуска программы открывается пустое окно, которое ожидает действий пользователя. При нажатии кнопки F1 создается треугольник с центром в точке 100, 100; для последующего перемещения данного треугольника нужно левой кнопкой мыши нажать в любое место окна и треугольник будет перемещен в указанную точку. Далее можно создать еще один треугольник, нажав на эту же клавишу (F1), либо создать новую фигуру, например, квадрат по нажатию клавиши F2, либо создать ромб по нажатию следующей кнопки F3. Стоит упомянуть, что каждая новая созданная фигура является активной и весь функционал применяется именно к ней, а предыдущая становится пассивной и последующие функции к ней не применяются. Так можно создавать до 10 фигур, видимых в окне, а также удалять и создавать новые взамен удаленных. Также программа предусматривает функционал отражения фигуры по 2 осям с помощью ПКМ, изначально отражение включено по вертикальной оси, а при нажатии пробела ось отражения меняется и также по нажатию ПКМ можно отразить фигуру, но только в горизонтальной оси. Еще одной функцией программы является увеличения и уменьшение активной фигуры с помощью клавиш “вверх” и “вниз”. Перемещаться между активными фигурами можно по нажатию клавиш “влево” и “вправо”. Для завершения работы программы требуется нажать на красную кнопку в левой верхней части окна.

# 4. Блок схемы алгоритма

Фреймворк Cocoa не имеет аналогов функции WinMain и WndProc, которые являются точкой входа для приложений и обработчиком сообщений окна соответственно, разработанных с использованием Win32 API.

Вместо этого, приложения на основе Cocoa используют функцию main, (взамен WinMain) которая объявляется в отдельном файле main.m и механизм делегатов и target-action (взамен WndProc). Эти функции определены в стандартной библиотеке Objective-C, включая приложения Cocoa. Однако суть работы очень схожа с функцией WinMain в WIN32 API, поэтому блок-схема к функции main будет выглядеть следующим образом:

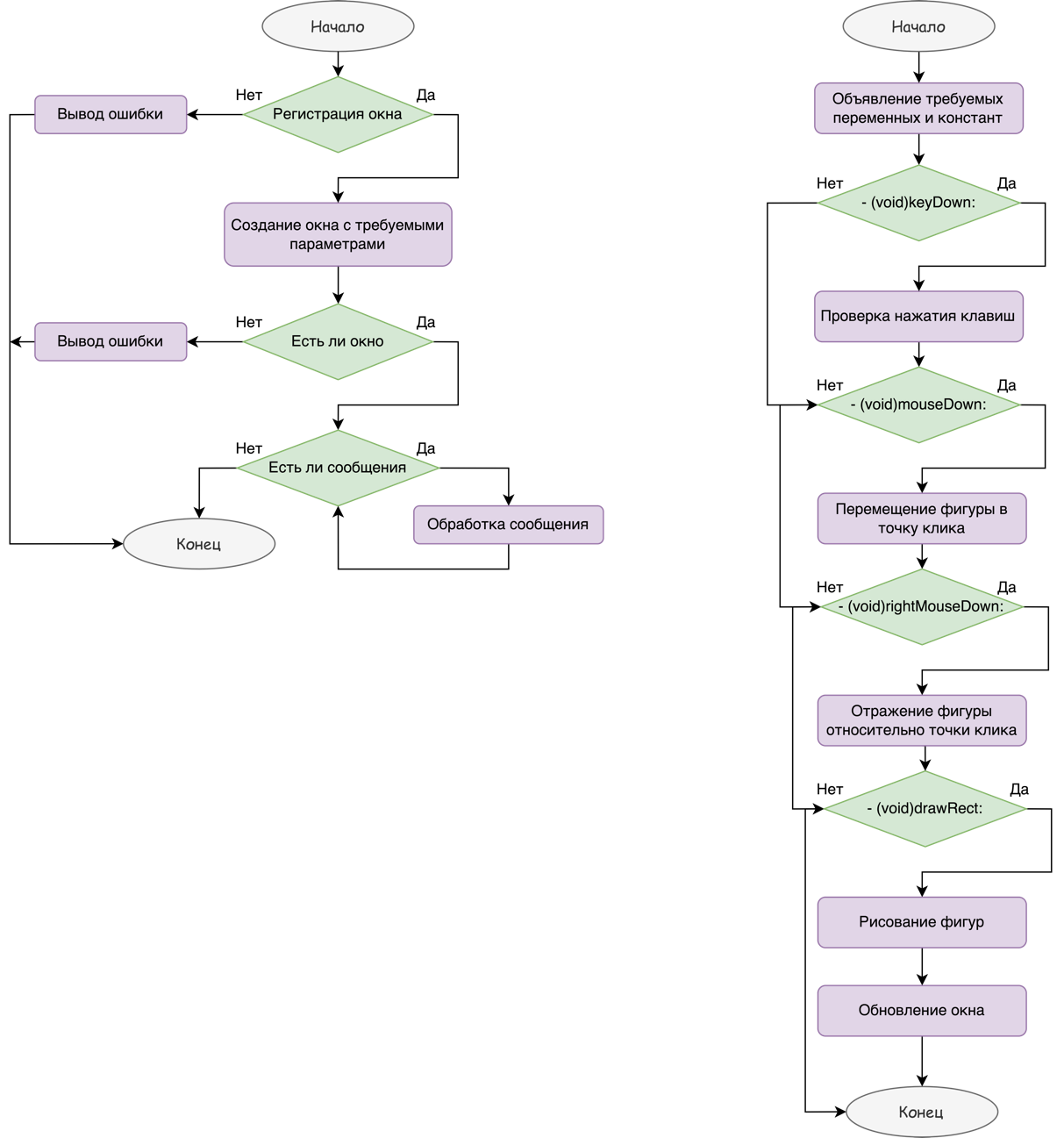


Рисунок 4 – Блок-схема функции main

Делегаты в Cocoa – это объекты, которые отвечают за реализацию определенного набора методов, связанных с событиями. Когда происходит событие, соответствующий объект делегата вызывается для обработки этого события. Следовательно блок схема к механизму делегатов будет выглядеть следующим образом:

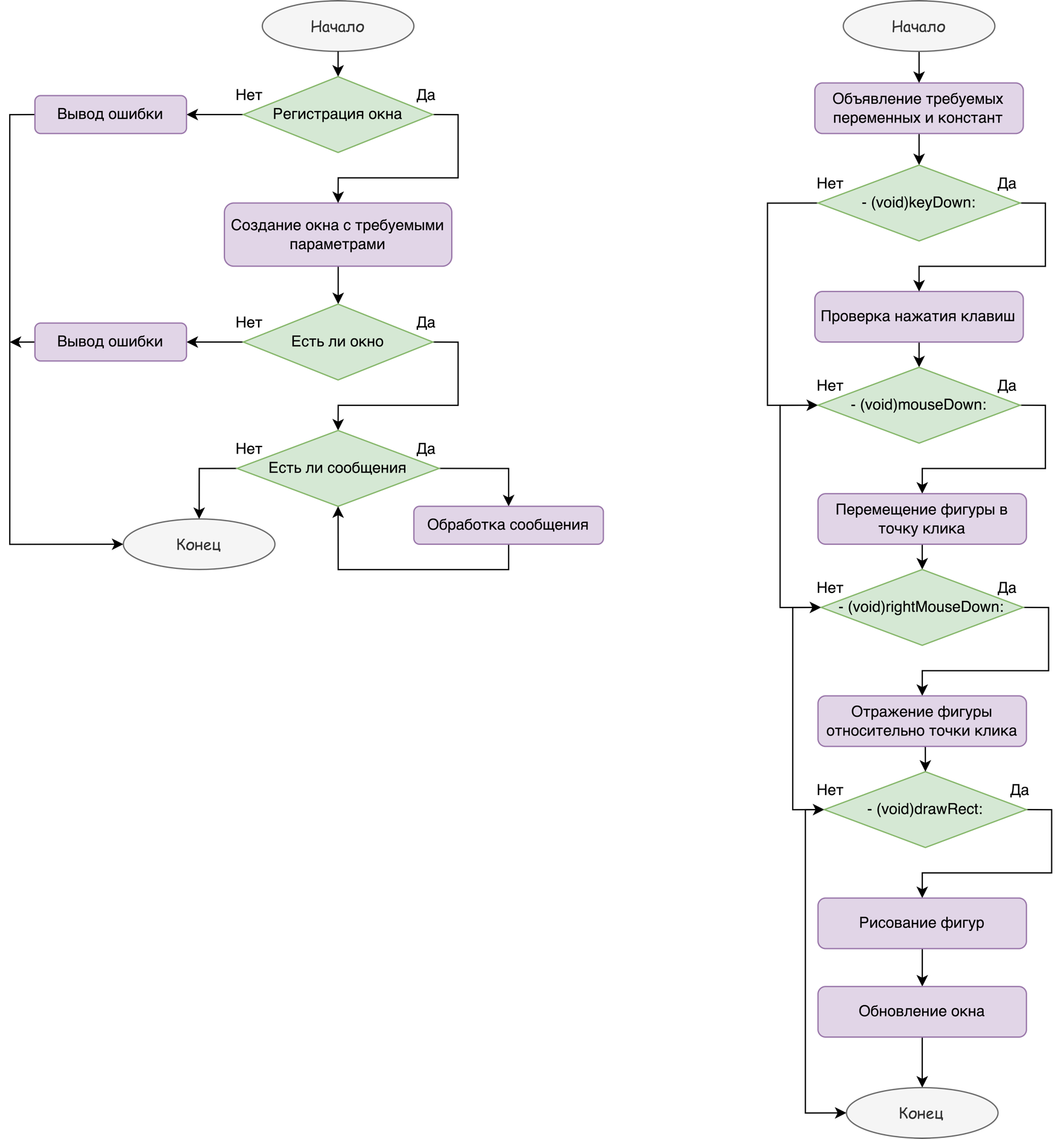


Рисунок 5 – блок-схема к механизму делегатов

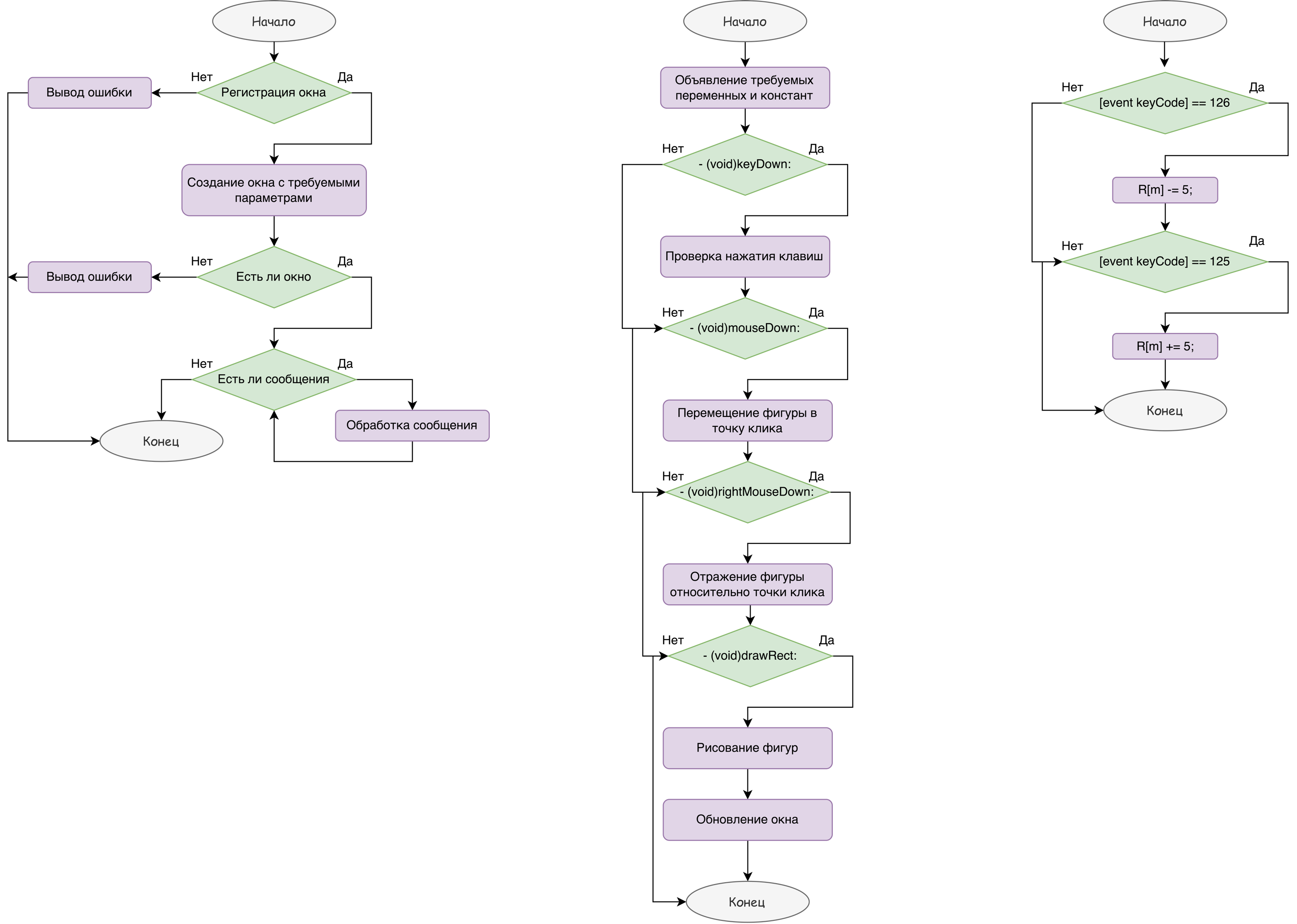


Рисунок 6 – блок-схема по выбору (изменение размера фигуры)

# 5. Список литературы

1. Scott Stevenson. Cocoa and Objective-C: Up and Running - United States of America. 2010. – 417с.

2. Michael Beam. Cocoa in a nutshell - United States of America. 2003. – 1581c.