СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Для того, чтобы познакомиться и с новым языком программирования и написать несложную программу, программисту необходимо ознакомиться с грамматическим описанием языка. Грамматическое описание любого языка программирования включает в себя алфавит, синтаксис и семантику.

Для описания правил синтаксиса языка программирования применяются формализованные системы обозначений – метаязыки. Существует два основных метаязыка:

- Расширенная форма Бэкуса-Наура (РБНФ);
- Синтаксические диаграммы.

Язык РБНФ более строг и точен, более удобен для представления синтаксиса в памяти машины, более компактен. Синтаксические диаграммы же более громоздки, однако намного нагляднее и проще для понимания.

Данная курсовая работа посвящена разработке программного средства для построения синтаксических диаграмм с использованием векторной графики.

В ходе выполнения данного проекта я постараюсь понять:

1. Как реализовать работу с графикой в языке программирования Delphi;
2. Как спроектировать наиболее удобный пользовательский интерфейс графического редактора;
3. Как устроен формат SVG;

В этой пояснительной записке отображены следующие этапы написания курсовой работы:

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;
2. Анализ требований к программному средству и разработка функциональных требований;
3. Проектирование программного средства;
4. Создание (конструирование) программного средства;
5. Тестирование, проверка работоспособности и анализ полученных результатов;
6. Руководство по установке и использованию.

1. АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

1.1. Описание синтаксиса языка с помощью синтаксических диаграмм

Синтаксическая диаграмма позволяет графически изобразить структуру синтаксической единицы.

В метаязыках, описывающих синтаксис языка программирования, используются следующие основные понятия:

- **Метапеременная –** обозначает определенную синтаксисом конструкцию языка. Для записи метапеременных в основном используются последовательности слов на естественном языке (русский, английский или др.) и служебных слов. Для разделения слов используется символ нижнего подчеркивания (_). В синтаксических диаграммах метапеременные заключаются в угловые скобки (<>). Метапеременная на размеченном ребре графа означает, что этот фрагмент диаграммы должен быть детализирован подстановкой синтаксической диаграммы с именем, соответствующим данной метапеременной.
  Примеры записи метапеременных:
  <Оператор_For>
  <Тип_Set>
  <Базовый_скалярный_тип>
- **Метаконстанты –** обозначает лексему языка программирования. В программе метаконстанте соответствует она сама. В синтаксических диаграммах метаконстанты записываются «как есть».
  Примеры метаконстант:
  For
  Begin
  Set
- **Метасимвол** – специальный символ, используемый для описания синтаксиса языка. В синтаксических диаграммах присутствует два единственных метасимвола:

- Метасимвол “::=” - используется для отделения имени синтаксической диаграммы.
- Метасимвол “<>” – используется для обозначения метапеременных

Синтаксическая диаграмма представляет собой ориентированный граф с размеченными ребрами. Для разметки ребер используются метаконстанты и метапеременные.

Представление в виде ориентированных графов основных конструкций:

1. Выбор (Альтернатива).

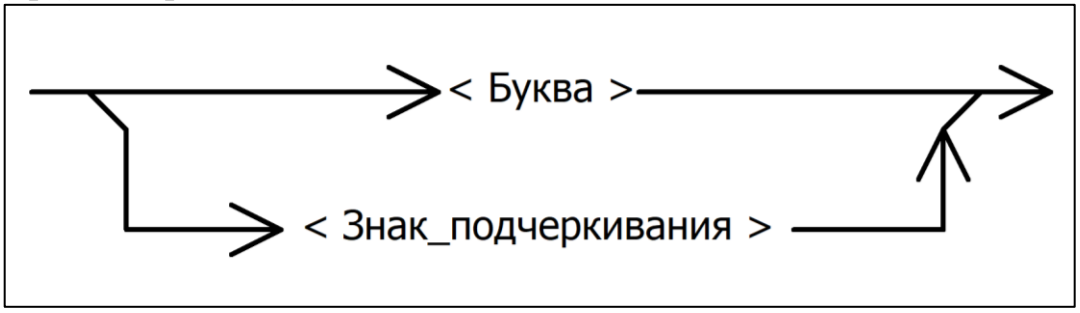

Рисунок 1.1 – пример конструкции выбора

2. Необязательная часть конструкции (Повторяется либо 1, либо 0 раз).

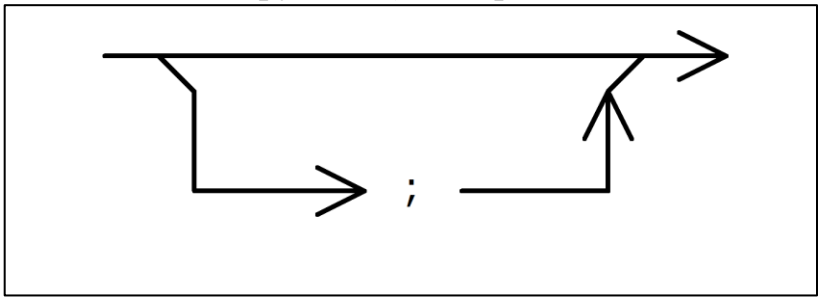

Рисунок 1.2 – пример необязательной части конструкции

3. Повторение конструкции

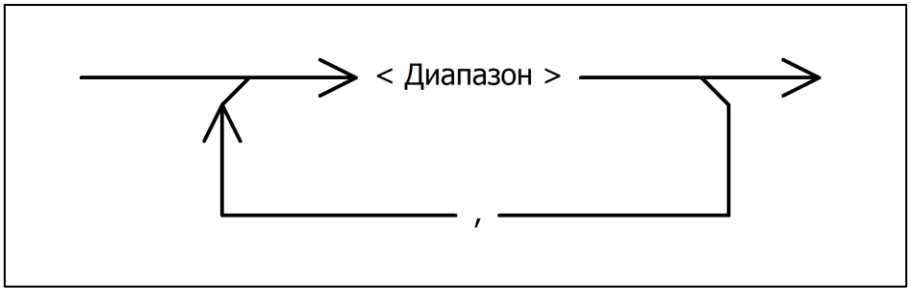

Рисунок 1.3 – пример повторения конструкции

1.2. Анализ существующих аналогов

После тщательных поисков мною не было найдено программного средства для создания синтаксических диаграмм, тем более, чтобы оно работало с векторной графики. Поэтому в качестве аналогов я рассмотрю графические редакторы с возможностью работы с векторной графикой.

Adobe Illustrator – один из наиболее распространенных векторных графических редакторов. Разрабатывается и распространяется компанией Adobe Systems.

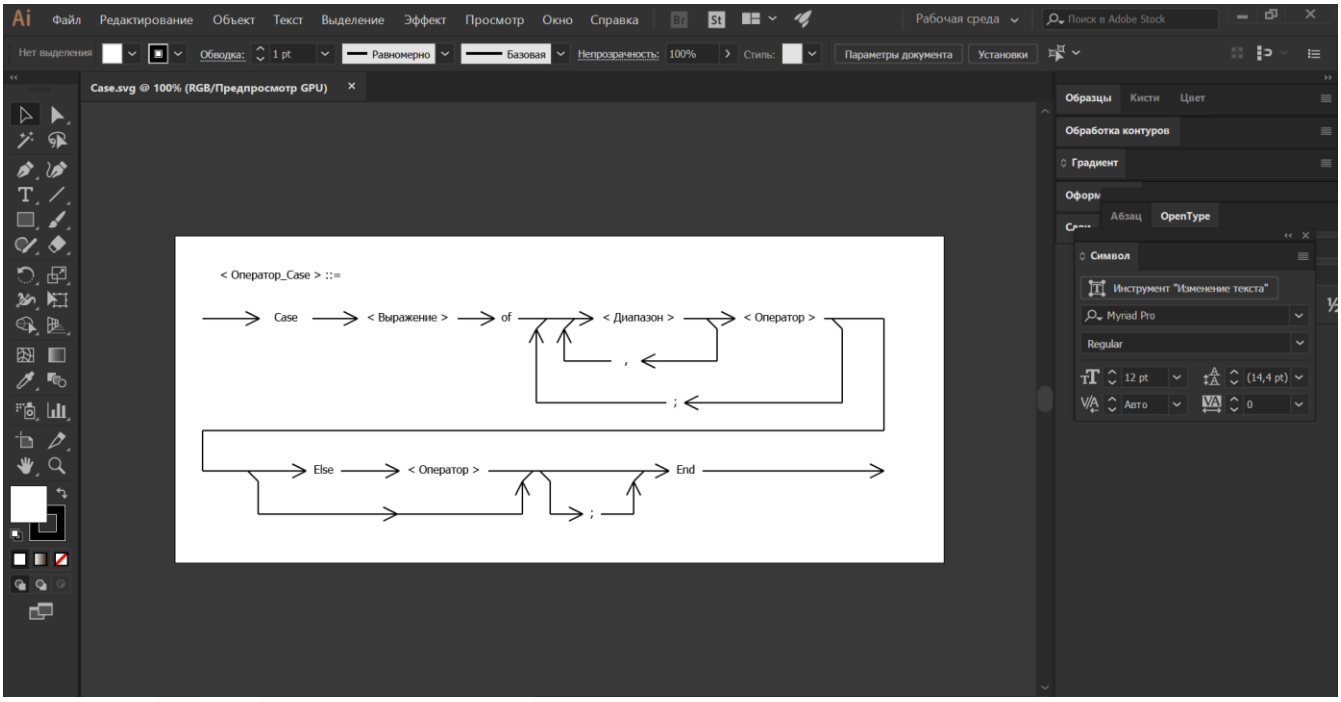

Рисунок 1.4 – Скриншот Adobe Illustrator

Плюсы:

- Удобный пользовательский интерфейс
- Стабильная работа программы
- Удобный процесс экспорта

Минусы:

- Высокая стоимость
- Программа заточена под редактирование произвольного векторного изображение, из-за чего есть много ненужных для построения синтаксических диаграмм функций, а также каждую стрелочку, линию, выравнивание фигур приходится производить самостоятельно.

CorelDRAW — графический редактор, разработанный канадской корпорацией Corel.

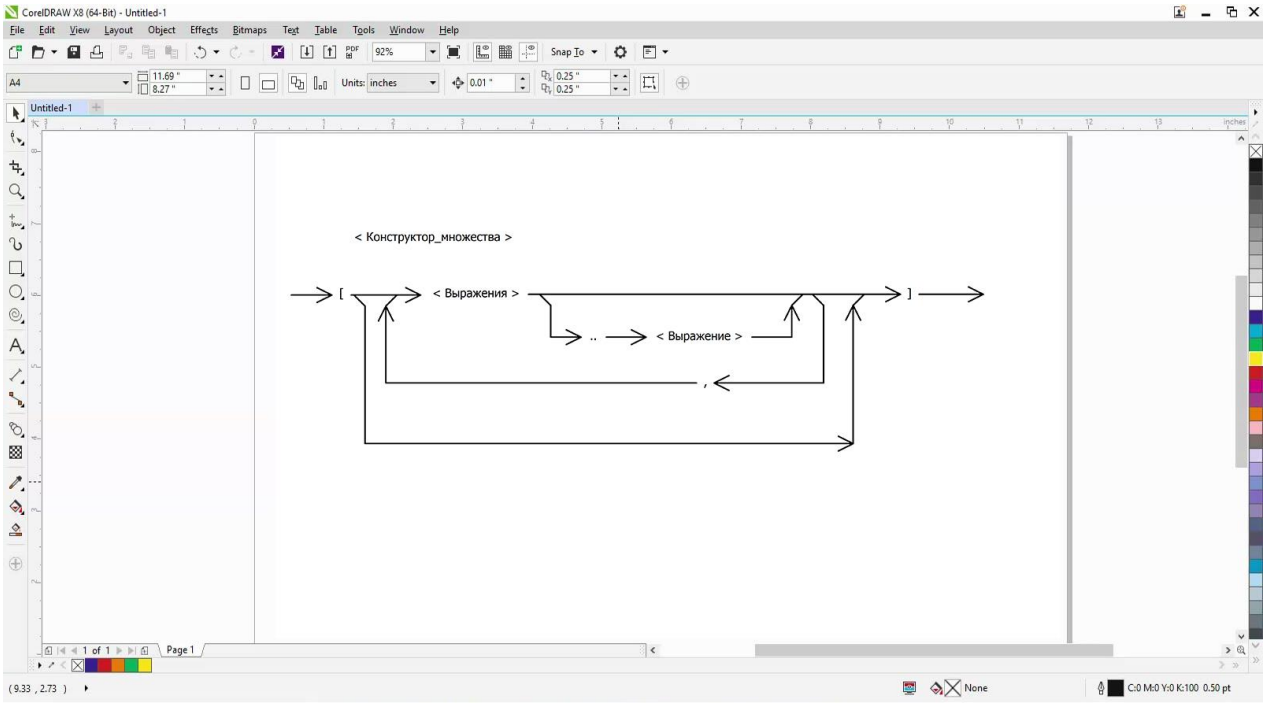

Рисунок 1.4 – Скриншот CorelDraw

Функционал CorelDraw очень схож с Adobe Illustrator.

Плюсы:

- Стабильная работа
- Удобный пользовательский интерфейс

Минусы:

- Высокая стоимость
- Программа не заточена под рисование синтаксических диаграмм

Inkscape – свободно распространяемый аналог Adobe Illustrator и Corel Draw. Работает с векторными изображениями в формате .svg.

Программа распространяется на условиях GNU (General Public License).

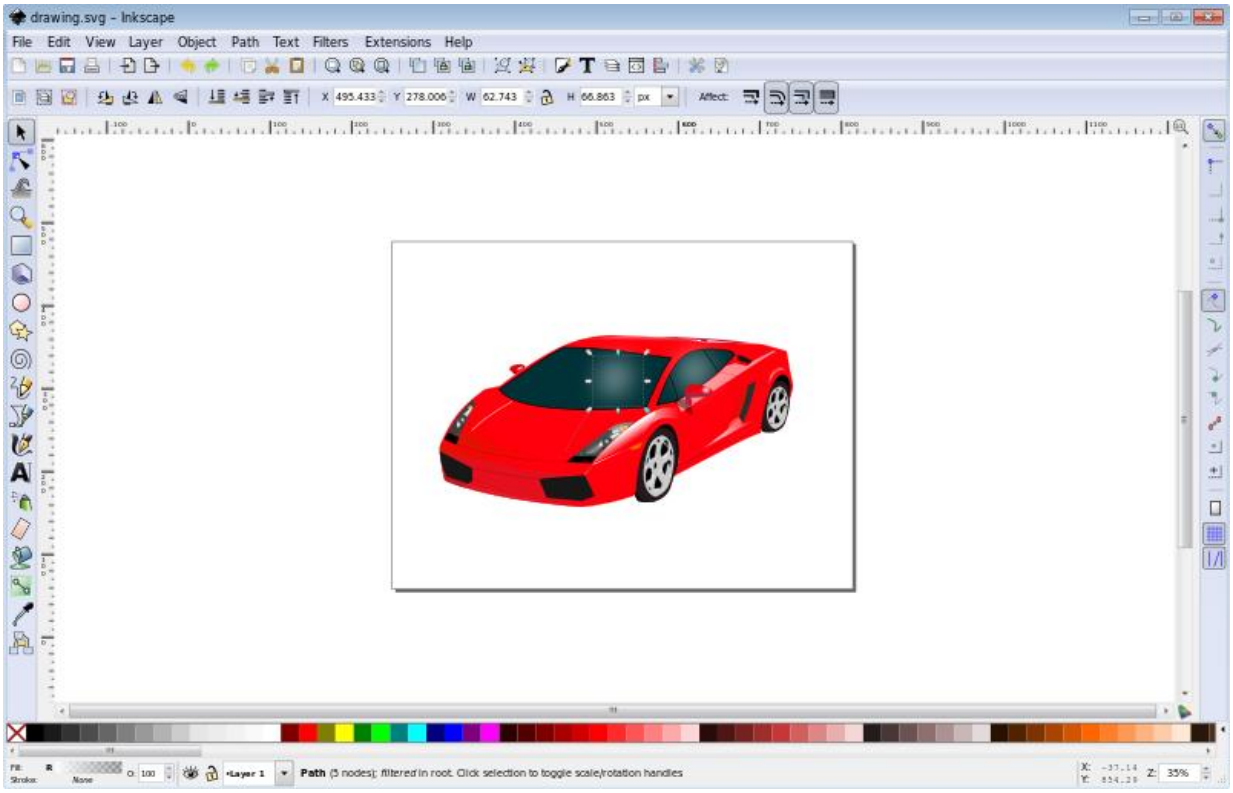

Рисунок 1.5 – скриншот inkscape

Плюсы:

- Бесплатный

Минусы:

- Программа не заточена под рисование синтаксических диаграмм
- Устаревший пользовательский интерфейс
- Повышенные требования к системным ресурсам
- «Сырость» ряда фильтров импорта
- Невозможность использования привычных горячих клавиш (например, Ctrl+C) если текущая раскладка не английская

1.3. Постановка задачи

Так как проектируемое программное средство должно быть заточено под создание синтаксических диаграмм, необходимо создать максимально удобный дизайн взаимодействия с пользователем (UX). Программа должна уметь сама рисовать мелкие элементы синтаксической диаграммы, такие как стрелки, диагональные скосы и т.д., основываясь на уже нарисованном пользователем изображении.

Чтобы программу можно было считать векторным редактором, должны быть реализованы следующие функции:

- Рисование с помощью установленных примитивных фигур;
- Изменение размеров, положения нарисованных фигур (редактирование изображения)
- Изменение размеров холста;
- Масштабирование изображение;
- Сохранение исходников изображения в типизированный файл с возможностью дальнейшего использования;
- Открытие типизированного файла с исходниками;
- Экспорт изображение в векторный формат (svg);
- Экспорт изображения в растровые форматы (bmp, png);
- Операции копирования, вставки и отмены действия.

Также в программе должна присутствовать справка, включающая в себе инструкцию по использованию, описание программы и т.д.

В качестве языка программирования выбран язык Delphi, так как программа курса предмета ОАиП включает в себя данный язык, а также потому, что я хорошо освоил язык, выполняя лабораторные работы.

2. МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1. Представление векторного изображения в формате SVG

Векторная графика – это способ представления изображений и объектов, основанный на математическом описании элементарный географических объектов.

Одним из самых распространенных форматов файлов векторной графики является формат SVG. Формат SVG предназначен для описание двумерной векторной и смешанной графики в текстовом формате XML.

Структура документа:

- Первая строка – стандартный XML заголовок с указанием версии, кодировки.
  Пример:
  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
- Вторая и третья строка – заголовок DOCTYPE, определяющий тип документа.
  Пример:
  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
- Четвертая строка – корневой элемент документ с указанием пространства имен SVG
  Пример:
  <svg version="1.1"
  baseProfile="full"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:ev="http://www.w3.org/2001/xml-events"
  width="100%" height="100%">
- Далее идет остальной текст документа, завершающийся закрытием тега </svg>

Отрисовка основных фигур:

- **Описание путей**

  Позволяет задать любую фигуру, описывая путь от начальной точки до конечной через промежуточные координаты. Строка с данными задается атрибутом d тега **path** и содержит команды, закодированные набором букв и чисел. Буквы – обозначают тип команды. Наиболее простые – M *(англ. moveto – переместить),* L *(англ. lineto – нарисовать линию).* Цифры, чаще всего, содержат координаты точек по осям X и Y.

  Пример: линия из точки (100,100) в точку (100,200)

  <path fill="none" stroke="black" d="M 100 100 L 100 200" />

- **Прямоугольник**

  Строка задается 4-мя основными атрибутами тега **rect**: координаты X,Y левой верхней точки (Атрибуты x, y), высота и ширина (Атрибуты height и width соответственно).

  Пример:

  <rect fill="white" x="400" y="600" width="300" height="200" />

- **Окружность**

  Строка задается 3-мя основными атрибутами тега **cricle**: координаты центра (Атрибуты cx, cy), радиус (Атрибут r).

  Пример:

  <circle cx="200px" cy="200px" r="104px" fill="red”/>

- **Вывод текст**

  Выводимый текст заключается в тег **text**, в котором в качестве атрибутов задаются свойства. Элементарные свойства для вывода текста – координаты левой верхней точки текста (атрибуты x, y).

  Пример:

  <text x="30" y="12" >Syntax Diagrams Editor</text>

Каждому тегу можно задать дополнительные свойства, описания и примеры которых находятся в документации по формату SVG.

2.2. Описание функциональности ПС

Программное средство должно представлять следующие способы взаимодействия с пользователем:

- Обработка нажатий мыши в области полотна для рисования.
- Обработка перемещения мыши в области полотна для рисования.
- Обработка нажатий клавиш на форме, в том числе обработка сочетаний клавиш.
- Обработка взаимодействия с элементами меню.
- Обработка изменения режима рисования.

Основу взаимодействия с пользователем для рисования представляет первый пункт – обработка нажатий мыши в области полотна для рисования. Взаимодействие может быть различно в зависимости от режима рисования.

Режим рисования может принимать один из следующих значений:

- Рисование прямоугольных фигур с текстом
  - Заголовок синтаксической диаграммы
  - Метапеременная
  - Метаконстанта
- Рисование линий
- Запрет рисования (Редактирование)

Если активирован режим рисования прямоугольных фигур с текстом, программное средство должно отобразить введенный пользователем текст на полотне по координатам клика.

Если активирован режим рисования линий, то первый клик левой кнопкой мыши по полотну должен активировать режим рисования линий. Каждое следующее нажатие левой кнопки мыши должно добавлять новую точку и соединить ее с предыдущей точкой данной линии. Нажатие правой кнопки мыши прекращает рисование линии.

Если активирован режим редактирования, курсор меняется в зависимости от того, на какую область фигуры он наведен:

- Вершина фигуры
- Сторона фигуры
- Центр фигуры

При зажатии мыши в этом режиме, фигура должна редактироваться по следующему принципу:

- Зажата в центре – при перемещении курсора перемещается вся фигура.
- Зажата на вершине – при перемещении курсора перемещается вершина и стороны, которым принадлежит вершина.
- Зажата сторона фигуры – при перемещении курсора перемещается сторона.

Клик по фигуре в любом из режимов должен выделять фигуру, по которой был произведен клик. При нажатии клавиши «delete» в любом из режимов должна удаляться выделенная фигура. Сочетания «Ctrl + C» должно помещать в специальный «программный буфер обмена» выделенную фигуру. Сочетания «Ctrl + V» - извлекать из буфера фигуру и отображать на полотне. Сочетание «Ctrl + Z» должно «откатывать» на предыдущее действие, то есть отменять последнее изменение. Если фигура (не линия) выделена, то если изменить текст в отведенном текстовом поле и нажать «enter», текст внутри фигуры должен измениться на введенный.

В зависимости от того, по какому элементу меню был произведен клик, программа должна уметь:

- Создавать новый файл.
- Сохранять исходный файл.
- Открывать исходный файл.
- Экспортировать в векторные и растровые форматы.
- Изменять размеры полотна.
- Включать/выключать режим «примагничивания».

2.3. Спецификация функциональных требований.

Среди функциональных требований есть «Отображение фигур на полотне».

Спецификация данной функции может иметь следующий вид:

- Прямоугольные фигуры имеют прозрачный фон и обводку, вершины обозначаются маленькими квадратами. Текст вписан в прямоугольник с вертикальным и горизонтальным центрированием.

- Линии должны проходить через все точки, заданные пользователем, в заданном пользователем порядке.
- В конце каждой линии проводится стрелка.
- Если первые две точки линии описывают вертикальный отрезок, то проводится дополнительный диагональный отрезок (см. рисунок 2.1).
- Если последние две точки линии описывают вертикальный отрезок, а перед этим присутствовал и горизонтальный участок линии, то проводится дополнительный диагональный отрезок (см. рисунок 2.2).
- Если линия начинается вертикальным участком, потом содержит горизонтальный участок и заканчивает вертикальным участком, по середине горизонтального участка ставится стрелка (см. рисунок 2.3).
- Если точка начала горизонтальной линии принадлежит вертикальному участку другой линии, то проводится дополнительный диагональный отрезок (см. рисунок 2.4).
- Если точка конца горизонтальной линии принадлежит вертикальному участку другой линии, то проводится дополнительный диагональный отрезок (см. рисунок 2.5.).
- Пользователь должен иметь возможность изменять размер полотна
- Пользователь должен иметь возможность изменить масштаб изображения

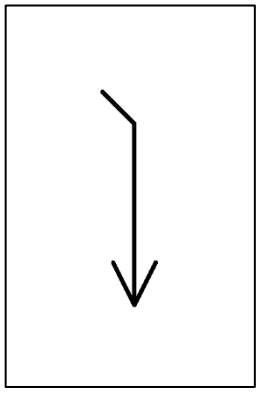

Рисунок 2.1 – дополнительный диагональный отрезок в начале вертикального участка линии.

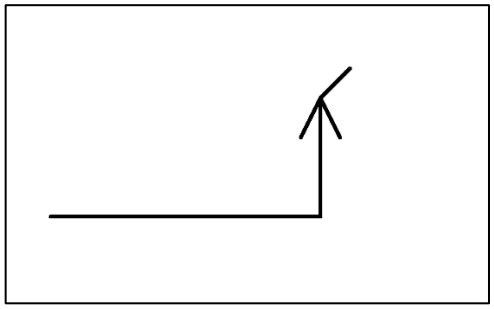

Рисунок 2.2 – дополнительный диагональный отрезок в конце вертикального участка линии.

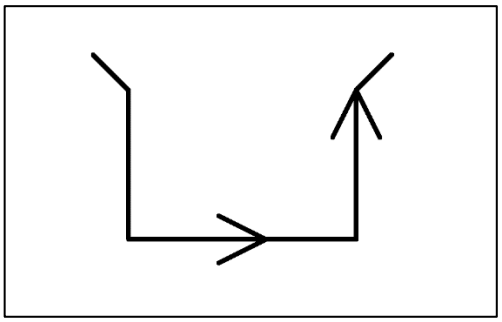

Рисунок 2.3. – стрелка по середине горизонтальной линии

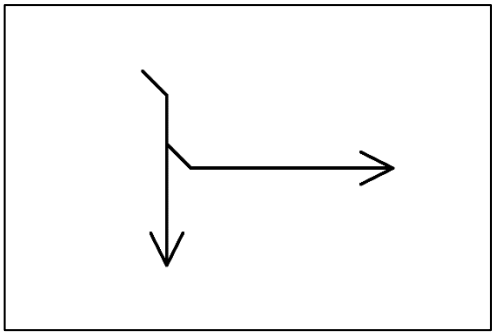

Рисунок 2.4. – дополнительный диагональный отрезок в начале горизонтального участка линии

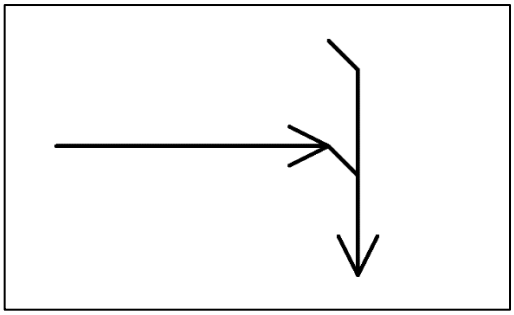

Рисунок 2.5. – дополнительный диагональный отрезок в конце горизонтального участка линии

3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1. Проектирование динамических структур данных

Для создания программного средства, для начала необходимо спроектировать, в каком виде будут храниться и взаимодействовать данные.

В первую очередь, необходимо хранить фигуры. Так как пользователь будет постоянно добавлять и удалять фигуры, напрашивается использование динамических списков. Однако нужно предусмотреть следующие особенности:

- Присутствуют различные фигуры с разным способом описания.
- У линий пользователь может динамически добавлять точки.
- Информативную часть нужно сохранять в файл, а в файле необходимо как-либо хранить сохраненное разрешение полотна и проверять файл на валидность.

Поэтому было принято решение использовать в качестве записи информативной части списка запись с вариантной частью. Структура представлена на рисунке 3.1.

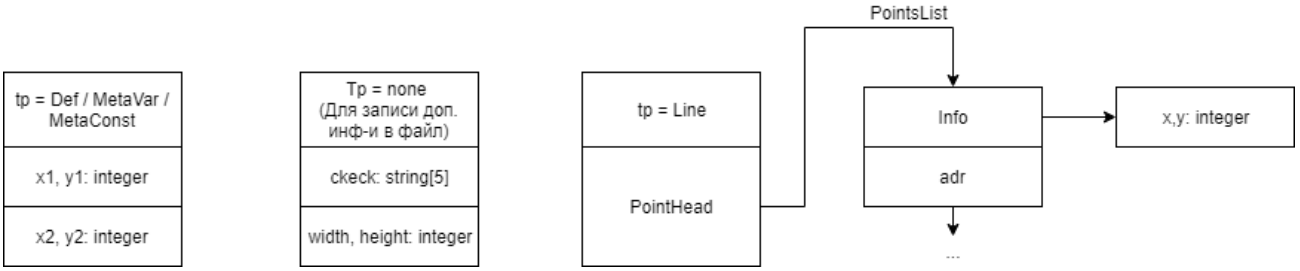

Рисунок 3.1 – структура основного списка программы

Так как фигура линии хранит в себе указатель на список точек линии, нужно придумать, как сохранять это в файл. Для этого я решил создать сделать отдельное представление записи для линий именно для сохранения в файл. При такой структуре координаты преобразуются в специальный текстовый формат, а при открытии файла, декодируются и преобразуются обратно – в список точек. Структура записи линии для записи в файл представлена на рисунке 3.2.

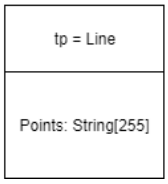

Рисунок 3.2. - структура записи линии для записи в файл

Для того, чтобы предусмотреть функционал отмены изменений, необходимо где-то хранить каждое изменение. Для этого лучше всего подходит стек, т.е. каждое изменение будет помещаться в вершину стека. При отмене изменений будет извлекаться вершина и вершина будет перемещаться к предыдущему элементу, то есть структура будет иметь вид, приведенный на рисунке 3.3.

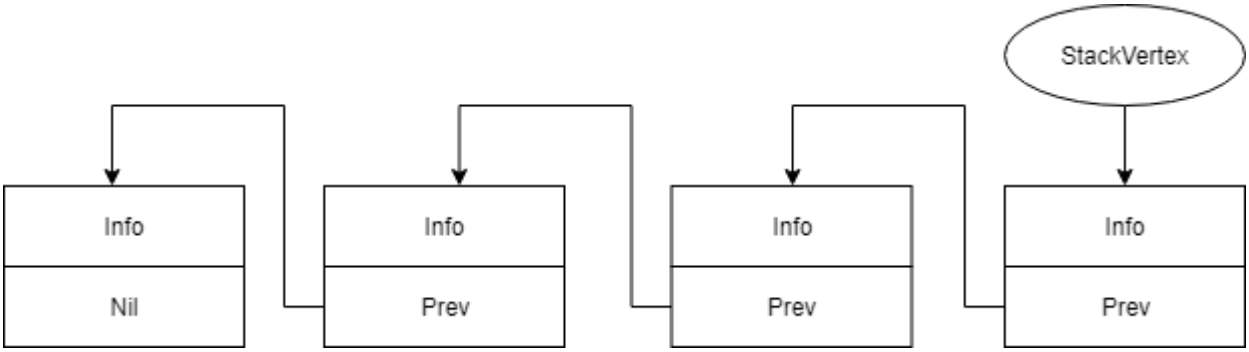

Рисунок 3.3 – структура стека изменений

Для того, чтобы не допустить извлечения самого последнего элемента, тем самым, потери указателя на вершину (т.к. он будет nil), я решил сделать переменную, отвечающую за «вариантность» информационной части стека специально типа и запретить изменять записи с таким типом.

Необходимо предусмотреть сохранение следующих изменений:

- Удаление фигуры
  Запись должна содержать:
  - Ссылку на удаленную фигуру
  - Ссылку на фигуру, расположенную в списке перед удаленной.
- Добавление точки линии:
  Запись должна содержать:
  - Ссылку на фигуру линии
  - Ссылку на добавленную точку в списке точек данной линии
- Добавление фигуры
  Запись должна содержать:
  - Ссылку на добавленную фигуру
- Перемещение фигуры/изменение размеров фигуры (кроме линий)
  Запись должна содержать:
  - Ссылку на фигуру, которую переместили
  - Предыдущее значение записи информативной части фигуры (предыдущие координаты)

- Перемещение линии / перемещение точки линии
  Запись должна содержать:
  - Ссылка на фигуру линии, которую переместили
  - Копию старых координат всех точек, сохраненных в текстовом формате (Используя тот же принцип, как при сохранении в файл).
- Изменение текста фигуры (Кроме линий)
  Запись должна содержать:
  - Ссылка на фигуру
  - Предыдущий текст
- Изменение размеров полотна
  Запись должна содержать:
  - Предыдущие значения ширины и высоты полотна

А также должна быть запись, сигнализирующая конец стека (используется только для единственной записи в конце стека и больше нигде).

После анализа тех изменений, которые необходимо предусмотреть, мною было принято решение вынести в общую часть информативной записи стека ссылку на фигуру, с которой произошли манипуляции, а остальное сохранять в вариантную часть.

3.2. Разработка алгоритма реакции на клик пользователя по полотну

После того, как пользователь кликнет по полотну, необходимо обработать данное событие, и, если текущий режим не является режимом редактирования, добавить фигуру. Также необходимо учесть, что должен быть фиксированный шаг «сетки» полотна для большего удобства пользователя. То есть, после клика координаты должны «округляться» до ближайшей вершины невидимой сетки. Схема алгоритма представлена на рисунке 3.4.

Режимы рисования и редактирования я решил записывать в отдельные перечислимые типы. (Для рисования: Рисование текстовой фигуры, рисование фигуры, нет рисования, для редактирования: перемещение фигуры, перемещение вершины, перемещение стороны).

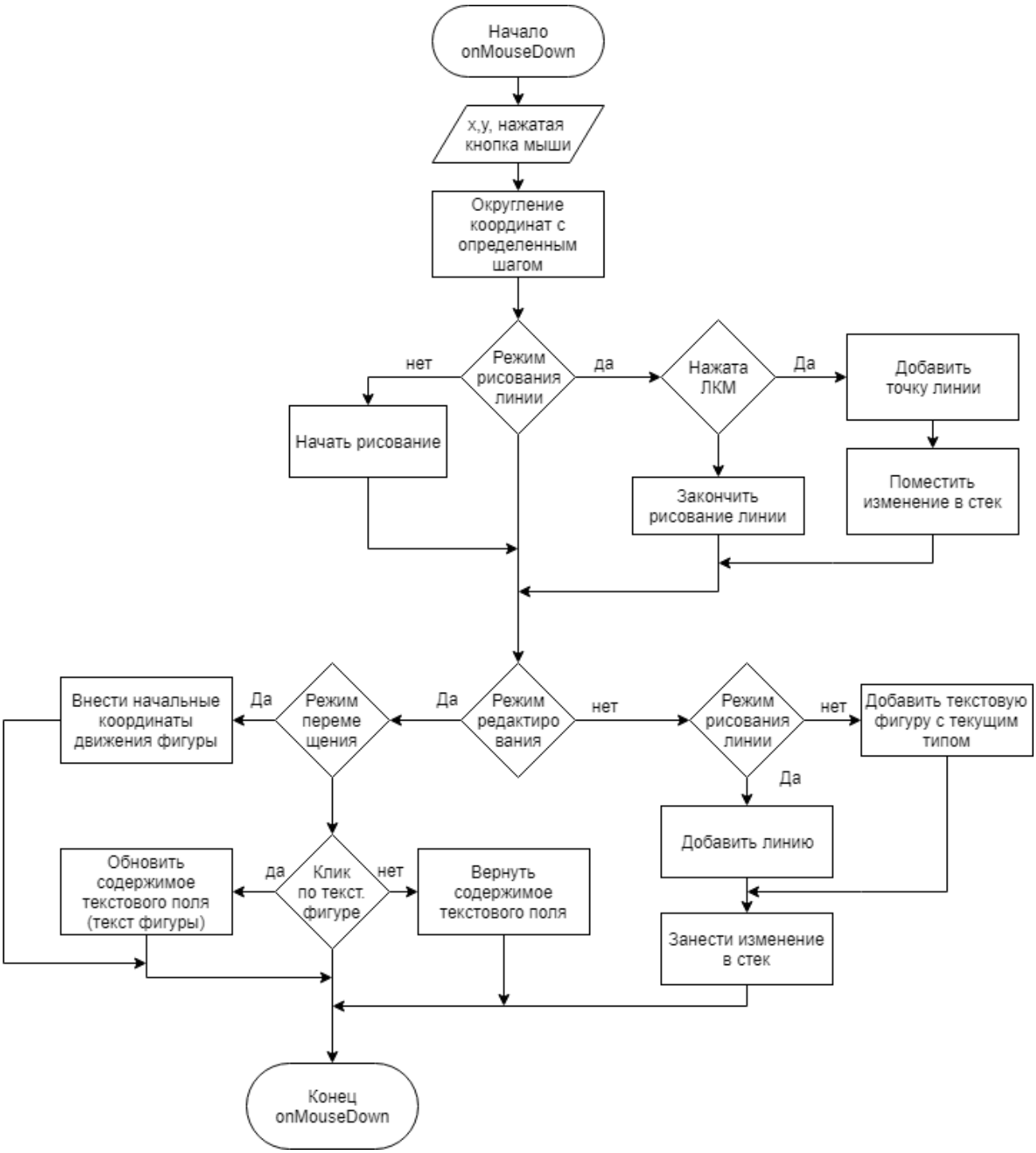

Рисунок 3.4 – схема алгоритма реакции на клик пользователя по полотну

3.3. Разработка алгоритма перемещение точки внутри линии

Если алгоритм перемещения фигуры / вершины текстовой фигуры / стороны текстовой фигуры / целой линии элементарен, из-за чего я решил даже не описывать его в данном разделе, то алгоритм перемещения одной точки прямой имеет «подводные камни». Если движется одна линия, может двигаться еще неограниченное количество точек, ведь линии всегда должны быть параллельны одной из осей. Чтобы избежать случаев, когда участок линии проходит под углом к обоим осям, необходимо разработать правильный алгоритм перемещения точки внутри линии.

Алгоритм должен принимать на вход ссылку на «голову» списка фигур, координаты до движения точки и координаты после движения точек. В процессе работы, алгоритм должен

3.4. Разработка алгоритма «примагничивания фигур»

Чтобы пользователю было проще редактировать синтаксические диаграммы, я решил добавить функционал «примагничивания» фигур.

Требования к алгоритму:

- Если начальная/конечная точка линии находится очень близко к другой линии, координаты этой точки должны измениться так, чтобы точка стала принадлежать прямой, расположенной очень близко.
- Если начальная/конечная точка линии находится очень близко к текстовой фигуре, координаты этой точки должны измениться так, чтобы точка оказалась (см. рисунок 3.5):
    - В центре прямоугольной фигуры по оси OY
    - На краю прямоугольной фигуры по оси OX
- Если текстовые фигуры располагаются приблизительно на одном уровне по оси OY, они должны выравниваться по одной координате Y.

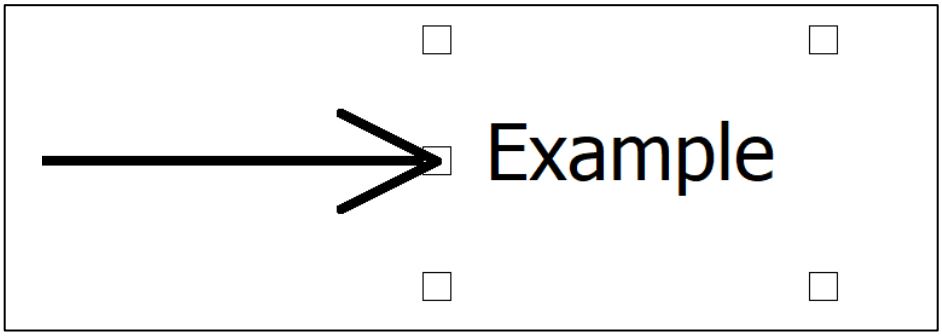

Рисунок 3.5 – пример положения точек после применения алгоритма «примагничивания» линии к текстовой фигуре.

В качестве значения, расстояние в сколько пикселей считать «близким», было принято вынести в отдельную константу для того, чтобы ее значение можно было изменить в любой момент.

3.4.1. Разработка алгоритма поиска линии вблизи точки

Так как линия представляет собой список точек, я решил вынести алгоритм поиска линии вблизи точки в отдельный алгоритм. Алгоритм должен на входе принимать ссылку на «голову» списка фигур и координату точки, вблизи которой надо найти линию. На выходе алгоритм должен вернуть точку, принадлежащую прямой, и которая будет наиболее близкая к исходной точке и ссылку на ближайшую точку списка точек прямой (nil если не найдено прямой поблизости).

Схема алгоритма представлена на рисунке 3.6.

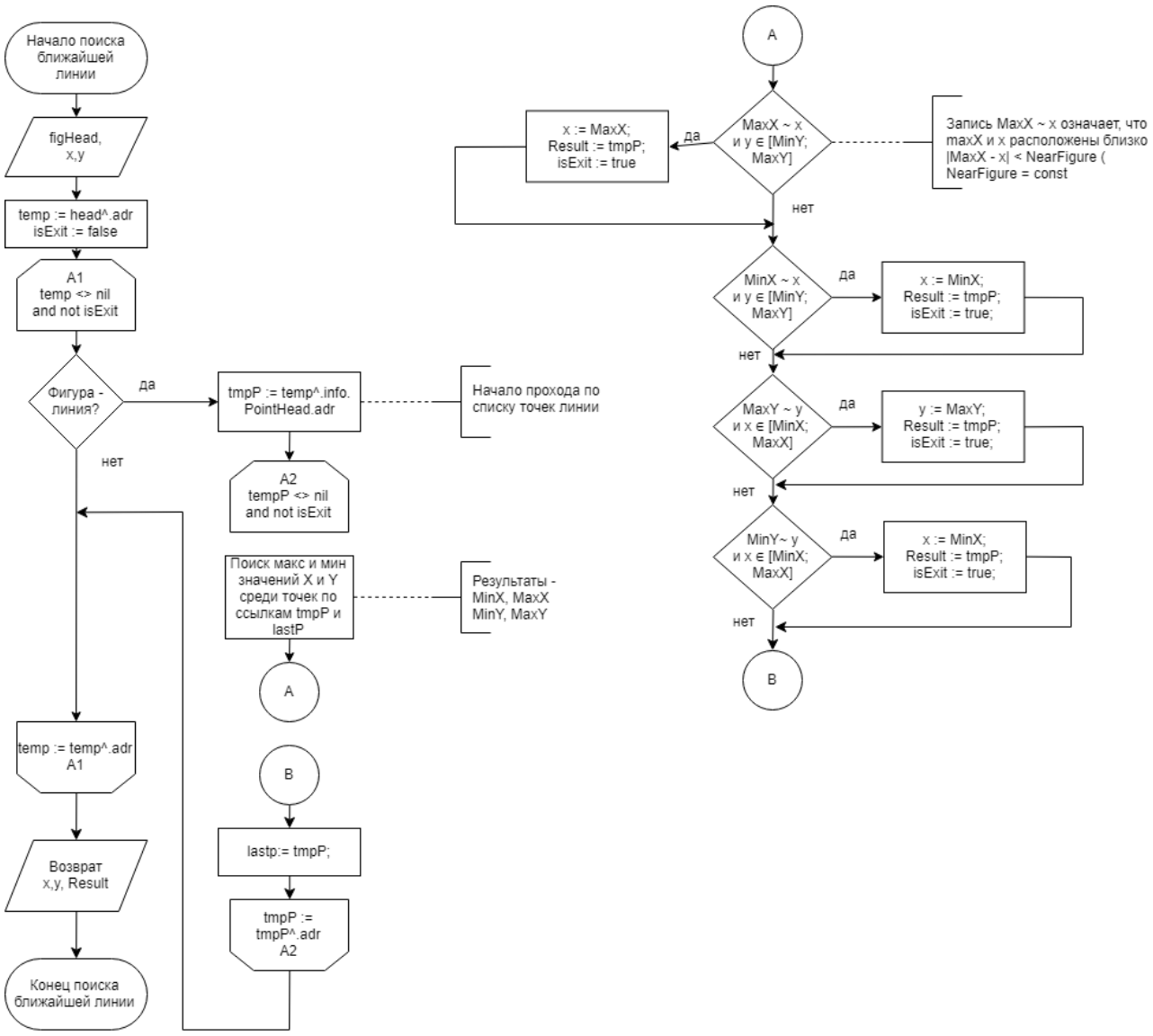

Рисунок 3.6 – схема алгоритма поиска ближайшей линии

3.4.2. Разработка алгоритма «примагничивания» линии к текстовой фигуре

Алгоритм должен принимать на входе ссылки на голову списка фигур и на точку определенной линии. Если алгоритм находит текстовую фигуру рядом с точкой, он меняет координаты этой точки и функция возвращает true. Иначе – алгоритм ничего не меняет, и функция возвращает false. Схема алгоритма представлена на рисунке 3.7.

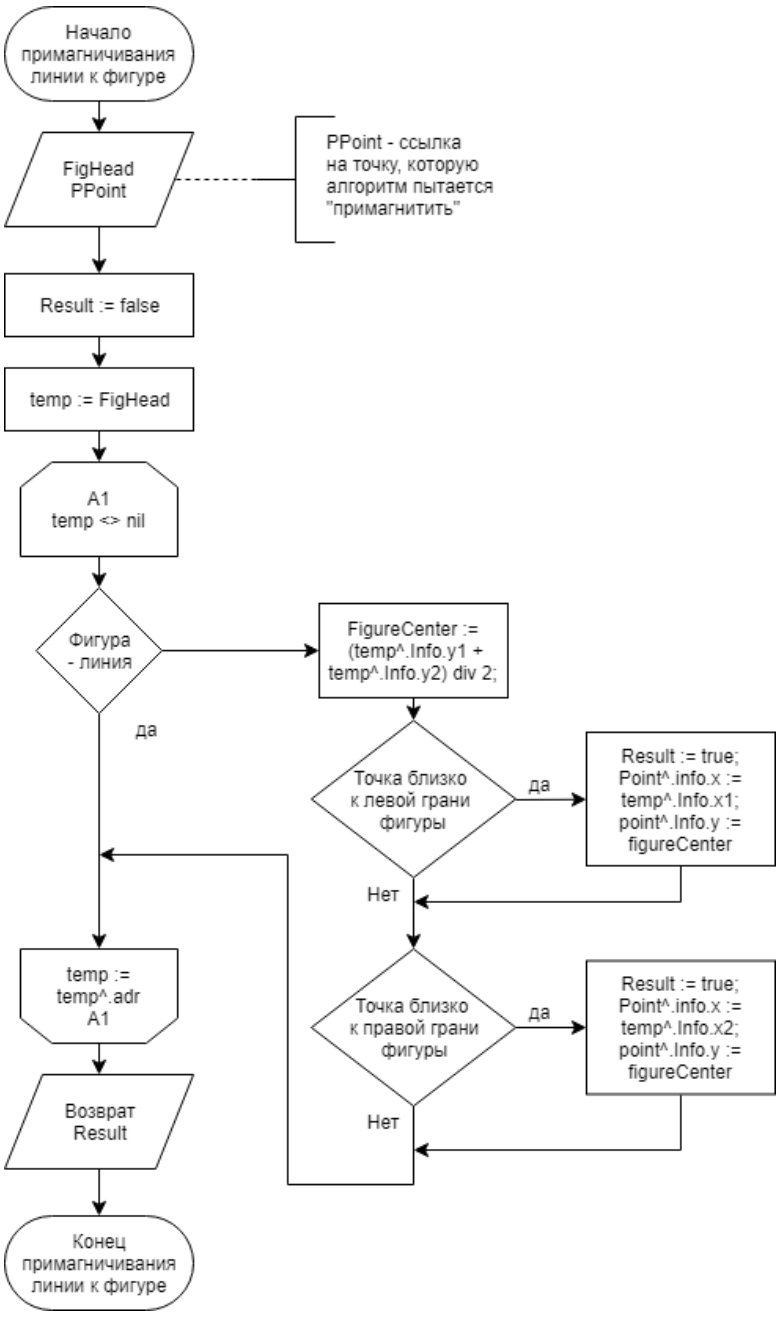

Рисунок 3.7 – Схема алгорита «примагничивания» линии к текстовой фигуре

3.4.3. Разработка алгоритма «примагничивания» точки к линии

Схема алгоритма представлена на рисунке 3.8.