## **СОДЕРЖАНИЕ**

[СОДЕРЖАНИЕ 4](#_Toc514747471)

[ВВЕДЕНИЕ 6](#_Toc514747472)

[1.1. Метод дамке представления алгоритмов 7](#_Toc514747473)

[1.2. Анализ существующих аналогов 9](#_Toc514747474)

[1.3. Постановка задачи 12](#_Toc514747475)

[2. МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 13](#_Toc514747476)

[2.1. Представление векторного изображения в формате SVG 13](#_Toc514747477)

[2.2. Описание функциональности ПС 15](#_Toc514747478)

[2.3. Спецификация функциональных требований 15](#_Toc514747479)

[3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 17](#_Toc514747480)

[3.1. Проектирование динамических структур данных 17](#_Toc514747481)

[3.2. Разработка алгоритма вставки фигуры 18](#_Toc514747482)

[3.3. Разработка алгоритма отрисовки фигур 20](#_Toc514747483)

[3.4. Разработка алгоритма получения координат по Y 21](#_Toc514747484)

[3.5. Разработка алгоритма сдвига блоков 22](#_Toc514747485)

[3.6. Разработка алгоритма вставки текста 23](#_Toc514747486)

[4. СОЗДАНИЕ (КОНСТРУИРОВАНИЕ) ПРОГРАММНОГО СРЕДСТВА 24](#_Toc514747487)

[4.1. Основной модуль программы 24](#_Toc514747488)

[4.2. Описание модуля Draw 26](#_Toc514747489)

[5.1. Описание модуля 28](#_Toc514747490)

[5.2. Описание модулей «Представления» 29](#_Toc514747491)

[6. ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 29](#_Toc514747492)

[6.1. Тестирование функционала добавления фигур 29](#_Toc514747493)

[6.2. Вывод из прохождения тестирования 29](#_Toc514747494)

[7. РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ 30](#_Toc514747495)

[ЗАКЛЮЧЕНИЕ 30](#_Toc514747496)

[СПИСОК ЛИТЕРАТУРЫ 30](#_Toc514747497)

[ПРИЛОЖЕНИЕ 1 31](#_Toc514747498)

[ПРИЛОЖЕНИЕ 2 31](#_Toc514747499)

# ВВЕДЕНИЕ

Задание алгоритмов с помощью блок-схем оказалось очень удобным средством изображения алгоритмов и получило широкое распространение. Блок-схема алгоритма — графическое изображение алгоритма в виде связанных между собой с помощью стрелок (линий перехода) и блоков — графических символов, каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия.

Многие специалисты в области теории программирования считают, что графическое представление алгоритмов в соответствии с *ГОСТ 19.701-90* скрывают структуру структурированной программы, представляя структурированность недостаточно очевидной. Поэтому для представления структурированных схем алгоритмов были предложены специальные методы графических обозначений.

Для описания алгоритма любой сложности используются графические схемы алгоритмов. Основными принято считать схемы по Госту, схемы по Дамке и диаграмы Наси-Шнайдермана.

К недостаткам ГОСТа можно отнести возможность чертить неструктурированные схемы алогритма. Данного недостатка лишены схемы по Дамке и диаграмы Наси-Шнайдермана.

Данная курсовая работа посвящена разработке программного средства для построения схем алгоритмов по Дамке.

В ходе выполнения данной работы будут рассмотрены и применены следующие подходы:

1. Работа с графикой в Delphi
2. Разработка понятного, удобного, а главное быстрого интерфейса графического редактора
3. Работа с различными форматами растрового и векторного изображения

В пояснительной записке отображены 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.
7. **АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ**

## Метод дамке представления алгоритмов

«Эта одна книга “М. Дамке «Операционные системы микроЭВМ»” (прим. автора) по факту - одна из немногих, которые вообще были выпущены и стали настольными книгами тех времён. Во времена CP/M не было сегодняшних интернетов, где бы могли её распространять. Ну и не было stackoverflow, где можно было бы почувствовать себя крутым программистом, не зная ничего о программировании. Так что способ по факту использовался не в одной книге. Просто ещё появились ISO/IEC и ГОСТ, а заодно стало модным кодить без проектирования и документирования в связи с наплывом в профессию школотронов.»

Оношко Д.Е.

М. Дамке предложил для конструкций структурированных схем алгоритмов специальные обозначения.

**Три основных конструкции**структурного программирования изображаются следующим образом:

1) ***Функциональный блок***по-прежнемуобозначается прямоугольником

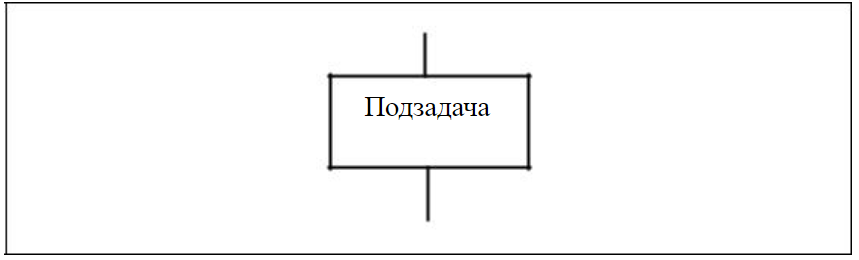


Рисунок 1.1 – Представление функционального блока по методу Дамке

2)***Конструкция If-Then-Else***изображается так, как иллюстрирует рисунок 1.2

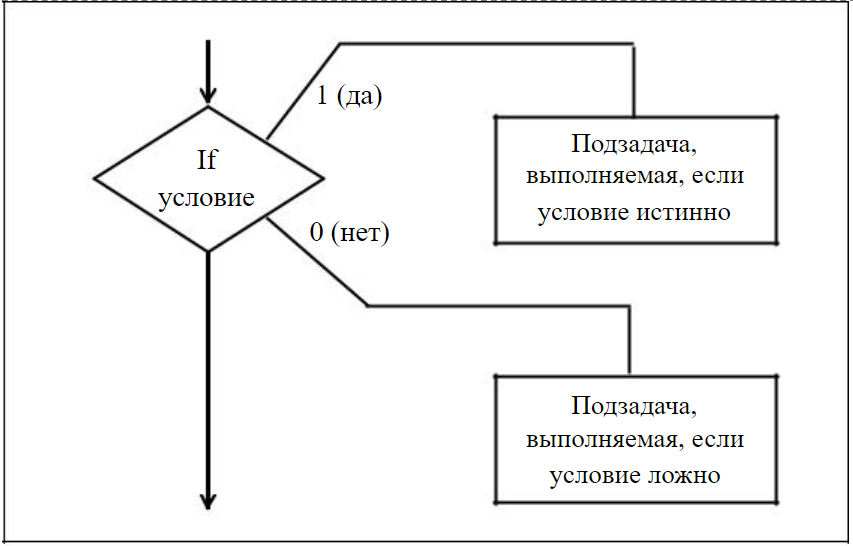
Элементы с выполняемыми действиями находятся справа от символа «Решение». Вход и выход из конструкции находятся соответственно сверху и снизу символя «Решение».

Рисунок 1.2 – Представление конструкции If-Then-Else по методу Дамке

3)***Конструкция Do-While(цикл с предусловием “Пока”)***изображается так, как показывает рисунок 1.3

Тело цикла выполняется до тех пор, пока условие истинно. Условие проверяется первым. Графически это изображается положением шестиугольника **над**выполняемым телом цикла.

Следует обратить внимание, что входы и выходы из всех конструкций метода Дамке находятся в левой части (сверху и снизу) графического представления конструкций. Расширения конструкций в правой части представления выходов не имеют.

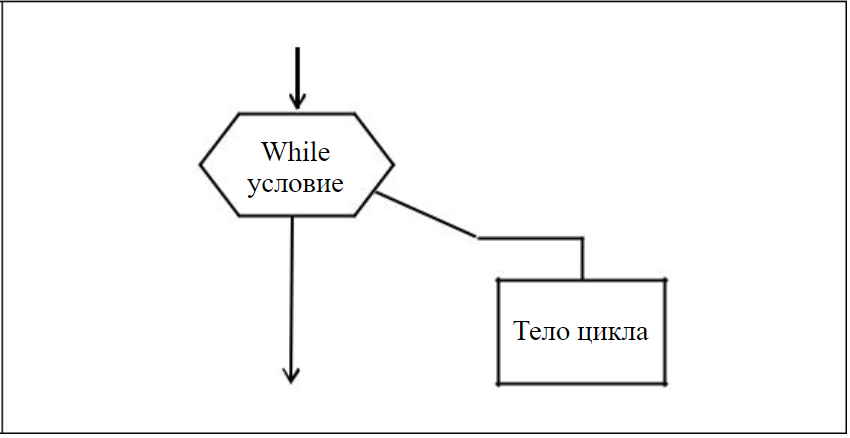


Рисунок 1.3 – Представление конструкции Do-While по методу Дамке

## Анализ существующих аналогов

В виду того, что рассматриваемый способ представления алгоритмов не пользуется популярностью, удобных программных средств для создания данных схем не существует. В качестве аналогов можно рассмотреть наиболее популярныег графические редакторы и сервисы.

**Draw.IO**

Один из самых удобных сервисов для разработки схем по Дамке. Разработчики - JGraph Ltd.

Плюсы:

* Не требует установки
* Стабильная работа сервиса
* Экспорт в большинство необходимых форматов

Минусы:

* Требует постоянного интернет-соединения
* Не предназначена для рисования схем по Дамке
* Бывают проблемы с открытием уже разработанных схем

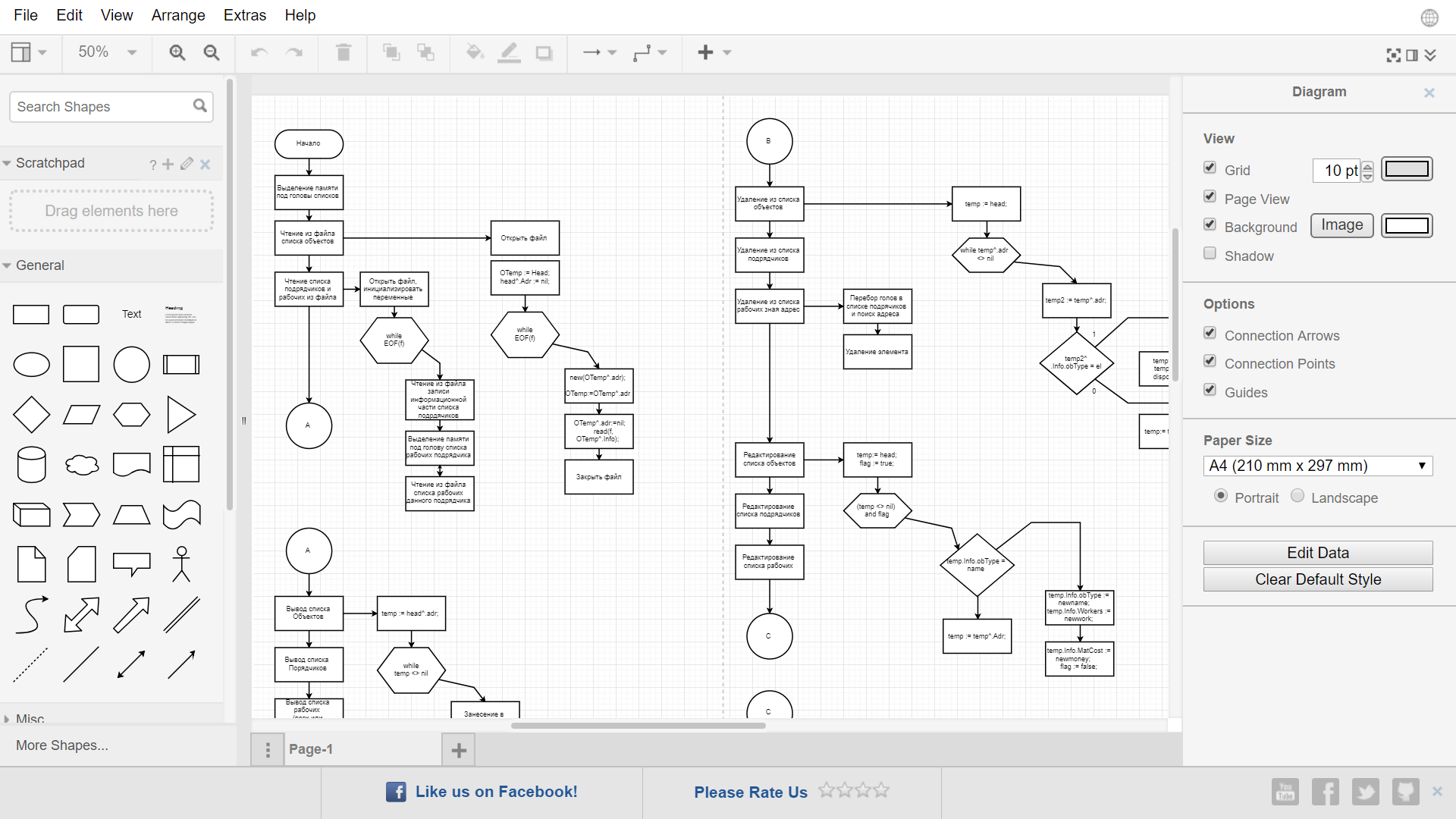


Рисунок 1.4 – Скриншот Draw.io

Популярный редактор Блок-схем от компании Gliffy Inc.

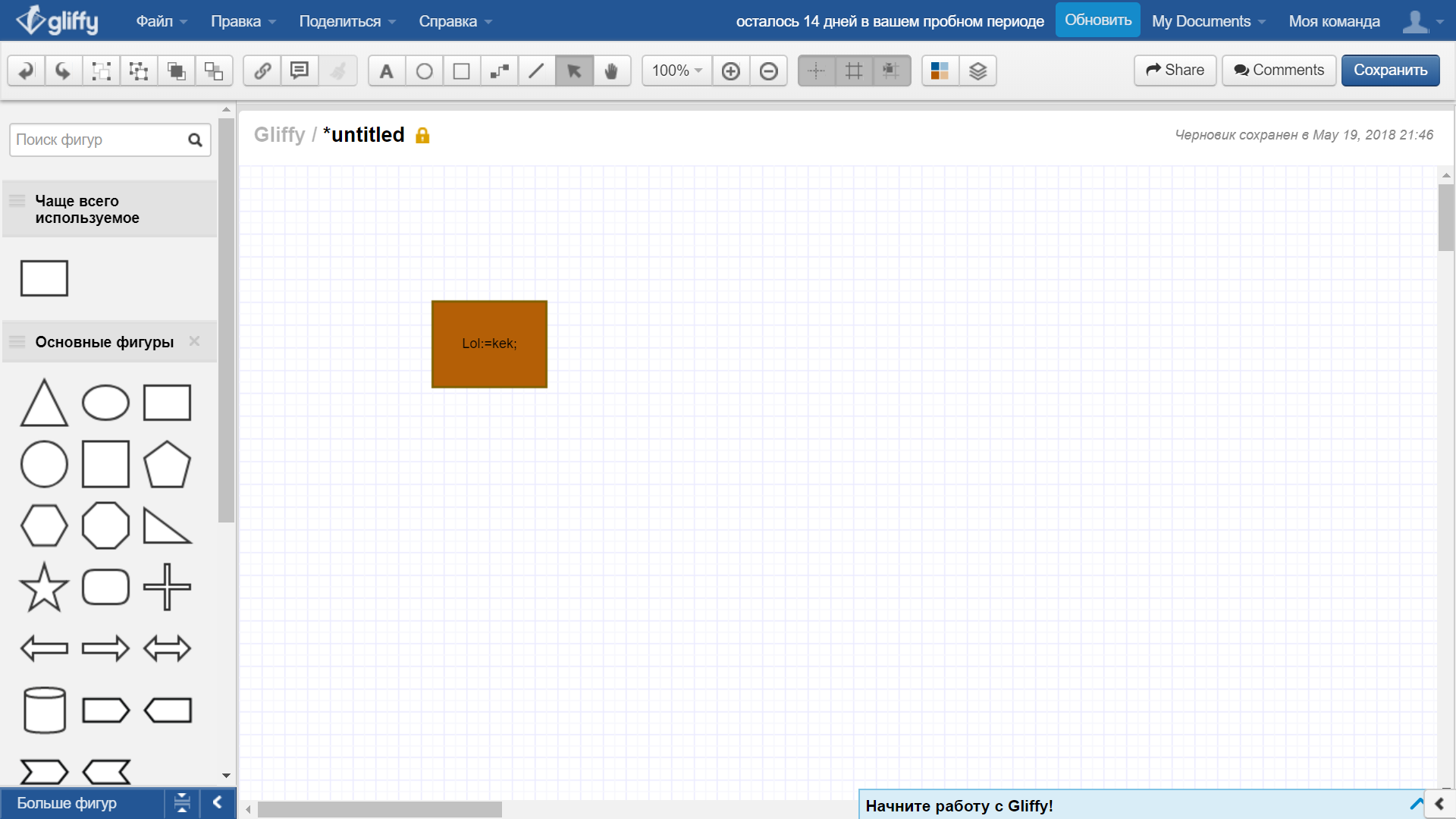


Рисунок 1.5 – Скриншот Gliffy.com

Плюсы:

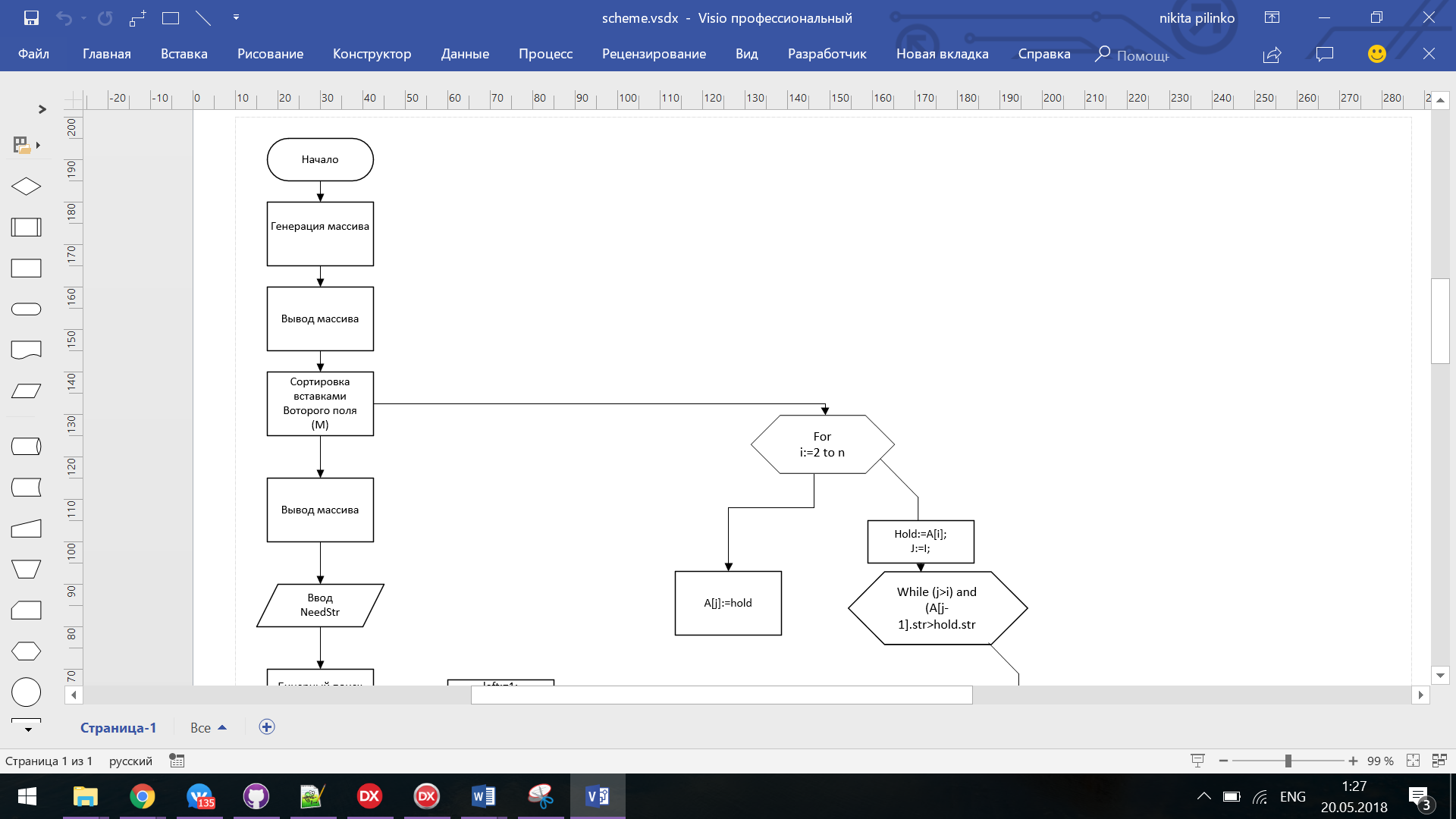
* Не требует установки
* Подробное руководство пользователя
* Экспорт в большинство необходимых форматов

Минусы:

* Требует постоянного интернет-соединения
* Не предназначена для рисования схем по Дамке
* Условно бесплатная – пробная версия (14 дней)

**MS Visio**

Стандартная программа из пакета Microsoft Office. Самая не пригодная к использованию программа для разработки схем по Дамке.



Плюсы:

* Большая палитра инструментов
* Автономная работа
* Экспорт в большинство необходимых форматов

Минусы:

* Не предназначена для рисования схем по Дамке
* Обладает интуитивно непонятным интерфейсом

## Постановка задачи

Так как проектируемое программное средство должно быть заточено под создание графических схем, необходимо создать максимально удобный дизайн взаимодействия с пользователем (UX). Программное средство должно автоматизированно прорисовывать основные конструкции и вести стрелки. Это должно положительно сказаться на скорости рисования схемы.

Так же будет реализована возмоность экспорта в векторные и растровые форматы изображения:

* Автоматизированное создание структуры данных, содержащей информацию для отрисовки блок схемы
* 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.
* Переключение языка (русский/английский)

Также в программе должно присутствовать руководство пользователя, включающее в себе инструкцию по использованию, описание программы и т.д.

В качестве языка программирования выбран язык Delphi.

# МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

## Представление векторного изображения в формате SVG

В отличие от растровых, векторные изображения состоят уже не из пикселей, а из множества опорных точек и соединяющих их кривых. Векторное изображение описывается математическими формулами и, соответственно, не требует наличия информации о каждом пикселе. Сколько ни увеличивай масштаб векторного изображения, вы никогда не увидите пикселей.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[2][3]:

* 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" />*

* **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.

## Описание функциональности ПС

Програмное средство должно обладать минмальным функционалом, достаточным для рисования схем по Дамке, но при этом обладать всеми необходимыми инструментами в том числе экспорт, настройка размеров холста.

Основные конструкции, необходимые для разработки схем по Дамке вынесены на ToolBar.

С самого запуска программы пользователю даётся возможность начинать работу. Блок начала алгоритма отрисовывается автоматически. Для навигации по блокам можно использовать стрелки, или явно указывать фигуру при помощи правого клика. Если на фигуре есть фокус, то по нажатию на соответсвующую кнопку ToolBarа будет отрисована выбранная конструкция. Для удаления блоков используется клавиша Delete. Так же реализована работа общепринятых клавишасочетаний Ctrl+Z, Ctrl+C, Ctrl+X, Ctrl+S.

Также программа предоставляет взаимодействие с меню В зависимости от того по какому элементу меню был произведен клик, программа может

* Создавать новый файл схемы
* Открывать существующий файл
* Экспортировать в векторные и растровые изображения
* Сохранять созданный файл в типизированный

## Спецификация функциональных требований

Основное функциональное требование - «Отображение фигур на полотне».

Спецификация данной функции может иметь следующий вид:

* Функциональный блок выглядит как Прямоугольник с прозрачным фоном и обводкой шириной в 3 px. Как и все остальные фигуры имеет цвет clBlack. Представлен на рисунке 2.1

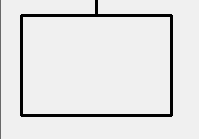


Рисунок 2.1 – Функциональный блок

* Конструкция If-Then-Else – самая сложная для отрисовки фигура, может иметь три выхода: при истинном и ложном значении и так же следующие за ней блоки.

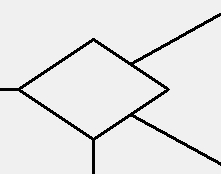


Рисунок 2.2 –конструкция If-Then-Else

* Конструкция Do – While, цикла с параметром и Repeat - until имеют схожее строение – это вытянутый по горизонтали шестиугольник, разница лишь в исходящих из блока линий – у Do – While она начинается из нижней левой границы как показано на рисунке 2.3, у Repeat – until из верхней левой.

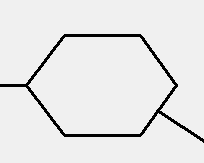


Рисунок 2.3 –конструкция Do-While

# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## Проектирование динамических структур данных

Поскольку память под данные отводятся во время выполнения работы программы, возникла необходимость использовать динамические структуры данных. Использование динамических структур позволяет организовать иерархию объектов, упрощает создание и удаление. Однако нужно предусмотреть следующие особенности:

* Функциональные блоки могут иметь различное количество дочерних компонентов
* Информативную часть нужно сохранять в файл, при этом сохранить связанность фигур между собой.

Поэтому было принято решение использовать в качестве записи ссылочной части списка запись с несколькими полями адреса. Структура представлена на рисунке 3.1.

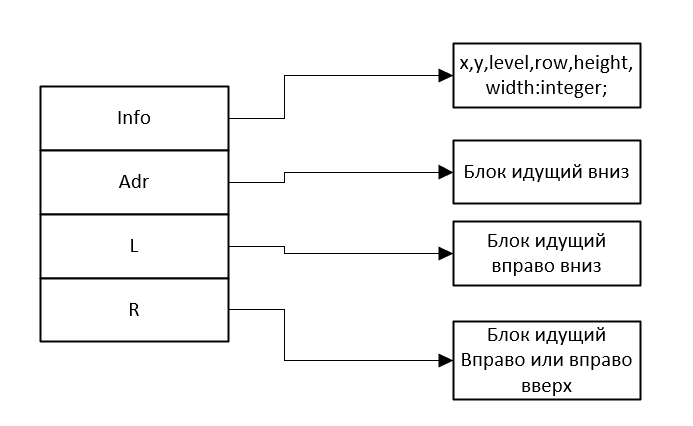


Рисунок 3.1 – структура основного списка программы

Так как фигуры могут иметь произвольную степень вложенности, то для прохода по структуре используются рекурсивные алгоритмы. Структура записи схемы алгоритма представлена на рисунке 3.2.

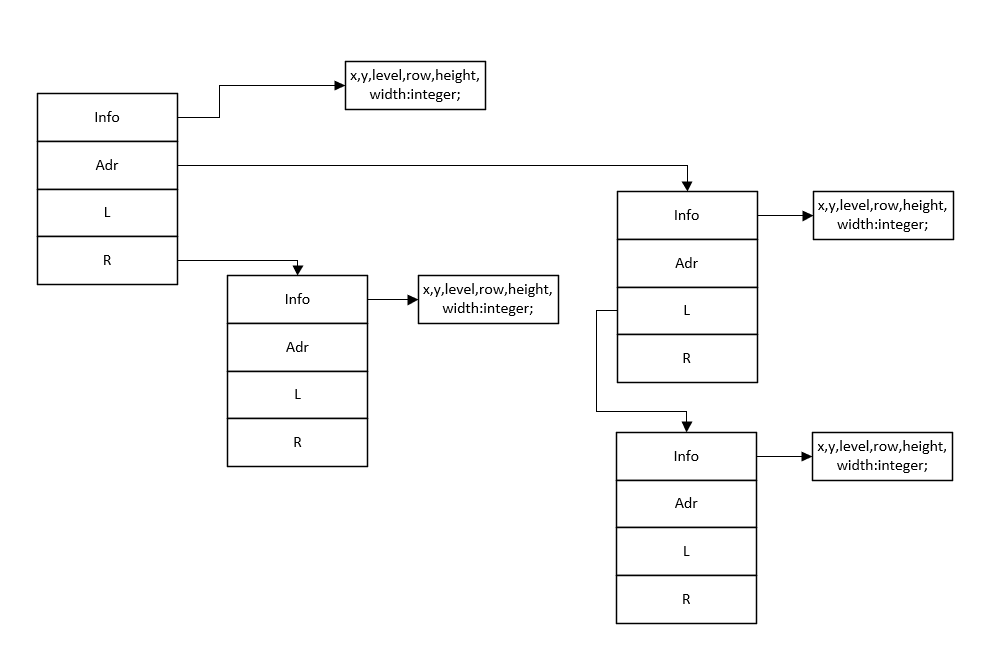


Рисунок 3.2. - структура схемы алгоритма

## Разработка алгоритма вставки фигуры

При активном фокусе на блоке, после нажатия на одну из кнопок ToolBara, управление передаётся процедуре, которой передаётся только тип фигуры. Процедура формирует запись типа TFigureinfo, содержащую всю необходимую информацию для формирования записи. Схема представлена на рисуноке 3.4

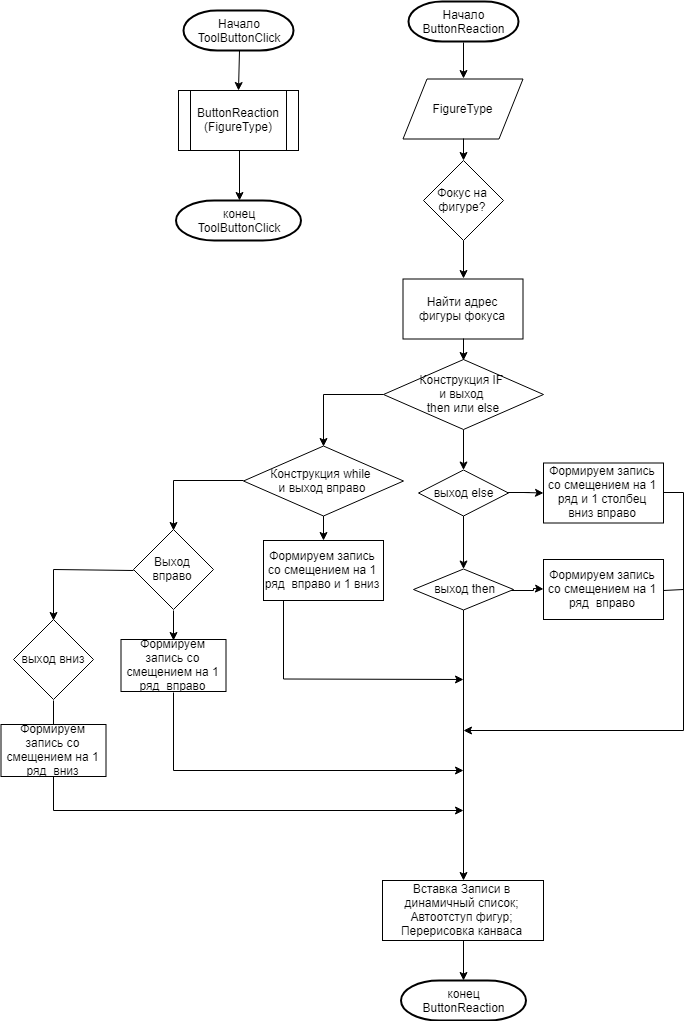


Рисунок 3.4 Реакция на нажатие

## Разработка алгоритма отрисовки фигур

Используемый подход отрисовки фигур имеет наименьшие затраты по памяти. Так как каждая запись содержит лишь координаты начала, высоту и ширину блока, тип блока, процедура вызывает внутри себя процедуры, которые отрисовывают необходимые фигуры внутри заданного прямоугольника. Метод Дамке предполагает произвольную степень вложенности блоков, необходимо использовать рекурсивный алгоритм прохода по записи. Схема алгоритма представлена на Рисуноке 3.5

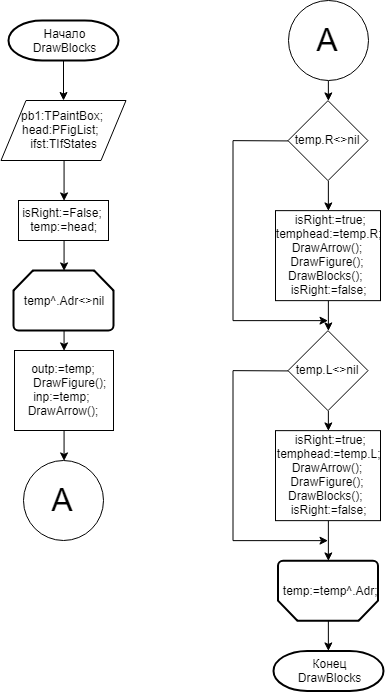


Рисунок 3.5 – схема алгоритма отрисовки

## Разработка алгоритма получения координат по Y

Чтобы предотвратить наложение фигур разной степени вложенности возникла необходимостьв разработки алгоритма, который бы позволил получать координаты свободного ряда. Функция, реализующая данный алгоритм вызывается каждый раз, когда блок вставляется вниз.

Основная идея состоит в определении Максимальной координаты Y блоков большего уровня вложенности. На основании полученного значения, и константы стандартного отсутпа очередная вставляемая фигура получает Y координату. Работа алгоритма представлена на рисунке 3.6а, 3.6б

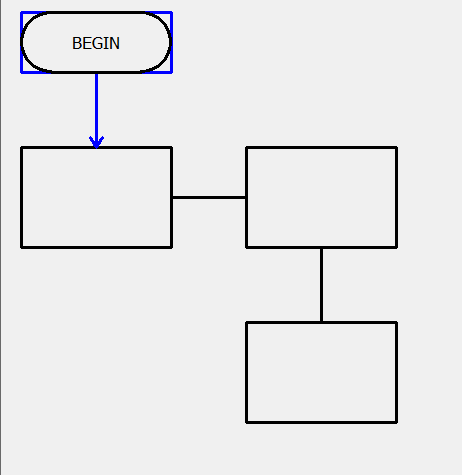
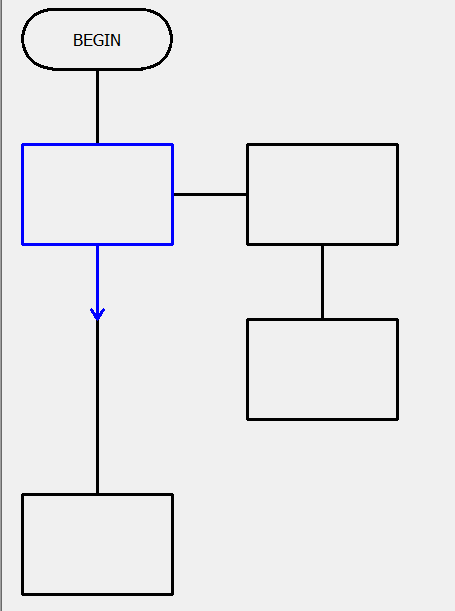
 

Рисунок 3.6а –исходная схема Рисунок 3.6б –результат работы алгоритма

Как видно из рисунков, применение такого алгоритма исключает возможность наложения блоков друг на друга

## Разработка алгоритма сдвига блоков

После вставки блока вложенность которого больше единицы так же может возникнуть ситуация, при которой блоки наложаться друг на друга. Для избежания этого был разработан алгоритм, который сдвигает фигуры, обладающие меньшей степенью вложенности. Так же меняется в соответствии со сдвигом поле ‘row’ сдвинутых блоков(Рисунок3.7)

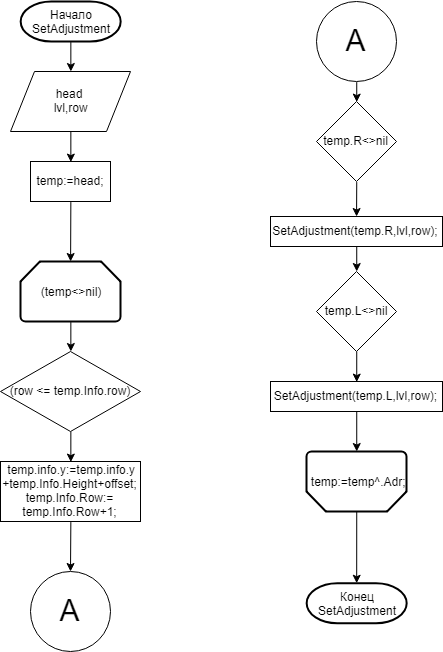


Рисунок 3.7 – схема алгоритма сдвига блоков

Рисунок 3.10 – Cхема алгоритма отрисовки линий

## Разработка алгоритма вставки текста

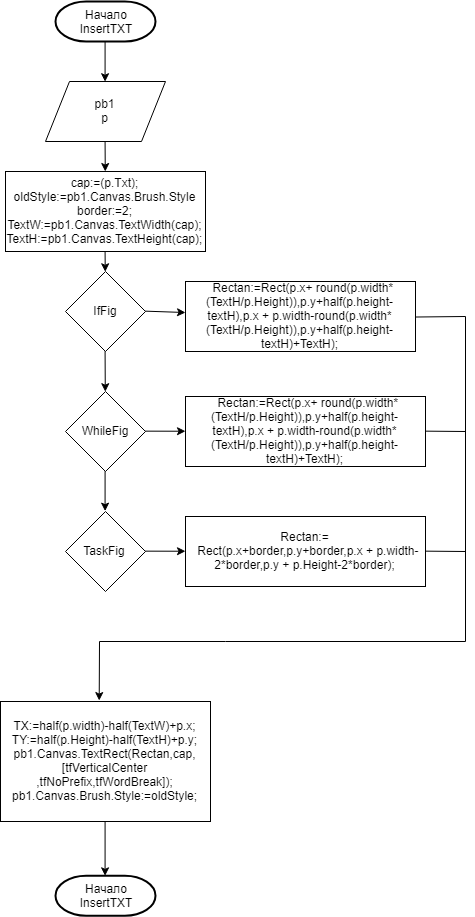
Текст должен вставляться по середине блока по вертилки и горизонтали. Для каждой фигуры область вставки текста различная. Процедура определяет границы области, при помощи определения параметров текста функциямиCanvas. Для конструкции IF-Then-Else область расчитывается по математической

Рисунок 3.8 – схема алгоритма вставки текста.

# СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА

## Разработка интерфейса программы

Интерфейс программы предельно простой, представляет собой главную форму с меню, tool bar и областью рисования.(Рисунок 4.1)

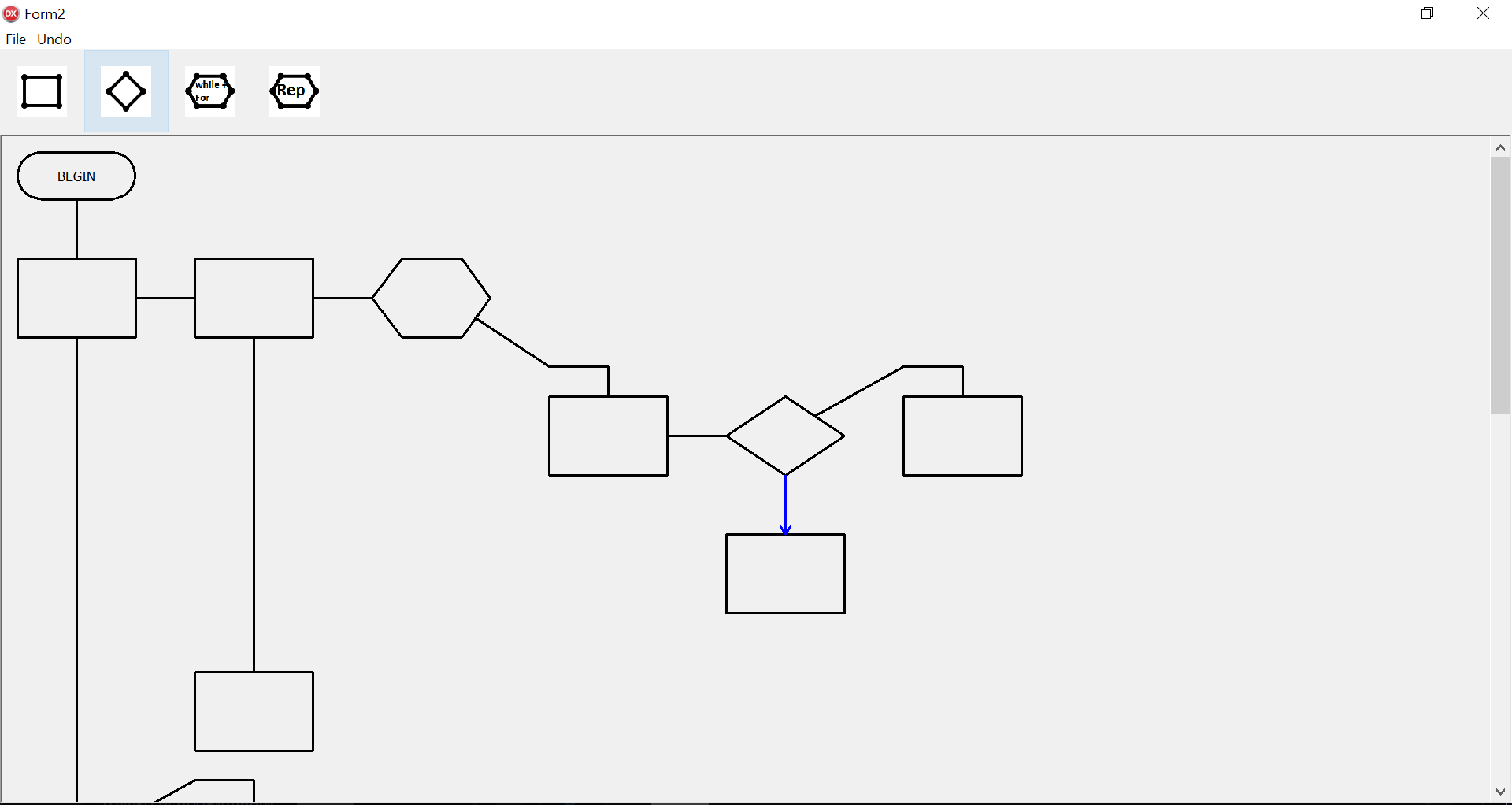


Рисунок 4.1

В форме использованы следующие элементы:

* TImagelist – Хранит упорядоченный набор иконок кнопок
* TMainMenu -Позволяет назначит меню с выпадающим списком
* TScrollBox – Подложка для виртуального расширения поля
* TMemo – многострочное поле для редактирования текста.
* TPaintBox - Область рисования
* TToolBar – Панель инструментов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя подпрограммы** | **Описание** | **Заголовок подпрограммы** | **Имя параметра** | **Назначение параметра** |
| clrscreen | $R \*.dfm | procedure clrscreen(pb1:TPaintBox); | pb1 |  |
| FormCreate | Событие при создании формы | procedure TKek.FormCreate(Sender: TObject); | Sender | Объект, который сгенерировал событие |
| getCharParams | Процедура получает ширину и высоту шрифта при помощи функций Windows | procedure TKek.getCharParams(var Chrwidth, Chrheight:Integer); | Chrwidth | Ширина |
| Chrheight | Высота |
| inptext | Процедура меняет свойство Visible TMemo, позволяет пользователю ввести текст | procedure TKek.inptext(pb1:TPaintBox; pinf:TFigureInfo); | pb1 | Объект на котором рисуем |
| pinf |  |
| FormKeyDown |  | procedure TKek.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState); | Sender | Объект, который сгенерировал событие |
| Key |  |
| Shift |  |
| pb1MouseDown | Обработка нажатие мыши | procedure TKek.pb1MouseDown(Sender: TObject; Button:TMouseButton; Shift: TShiftState; X, Y: Integer); | Sender | Объект, который сгенерировал событие |
| Button | кнопка |
| Shift |  |
| X |  |
| Y |  |
| ButtonReaction | Процедура | procedure ButtonReaction(Figure: TFigType); | Figure |  |
| pb1Paint | Процедура отрисовки канваса | procedure TKek.pb1Paint(Sender: TObject); | Sender | Объект, который сгенерировал событие |
| btntaskClick | Обработка клика, вызывает ButtonReaction | procedure TKek.btntaskClick(Sender: TObject); | Sender | Объект, который сгенерировал событие |
| btnWhileClick | Обработка клика, вызывает ButtonReaction | procedure TKek.btnWhileClick(Sender: TObject); | Sender | Объект, который сгенерировал событие |
| btnIfClick | Обработка клика, вызывает ButtonReaction | procedure TKek.btnIfClick(Sender: TObject); | Sender | Объект, который сгенерировал событие |
| ScrollBox1MouseWheelDown | Обработка прокрутки | procedure TKek.ScrollBox1MouseWheelDown(Sender: TObject;Shift: TShiftState; MousePos: TPoint; var Handled: Boolean); | Sender | Объект, который сгенерировал событие |
| Shift |  |
| MousePos |  |
| Handled |  |
| ScrollBox1MouseWheelUp | Обработка прокрутки | procedure TKek.ScrollBox1MouseWheelUp(Sender: TObject; Shift:TShiftState; MousePos: TPoint; var Handled: Boolean); | Sender | Объект, который сгенерировал событие |
| Shift |  |
| MousePos |  |
| Handled |  |
| ToolButton1Click | Обработка клика, вызывает ButtonReaction | procedure TKek.ToolButton1Click(Sender: TObject); | Sender | Объект, который сгенерировал событие |

Основное назначение модуля – обработка событий, которые генерирует пользователь.

## Описание модуля Draw

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. **Имя подпрограммы** | **Описание** | **Заголовок подпрограммы** | **Имя параметра** | **Назначение параметра** |
| TextUtil | Определяет длинну введённой строки | function TextUtil(cap:string):string; | cap |  |
| InsertTXT | Процедура, отрисовывающая текст на  Канвасе | procedure InsertTXT(pb1:TPaintBox;var p:TFigureInfo); | pb1 | Объект, который сгенерировал событие |
| getCharParams | Получает ширину и высоту символов | procedure TKek.getCharParams(var Chrwidth, Chrheight:Integer); | p | Ширина |
| IsEmptyTXT | Возвращает true если запись не содержит текста. | function IsEmptyTXT(p:TFigureInfo):Boolean; | p | Высота |
| HorizontalAdjust | Выравнивает блоки п огоризонтали | procedure HorizontalAdjust(head: PFigList; NewSize:PFigList); | head | Голова |
| NewSize |  |
| drawRect | Рисует прямоугольник | procedure drawRect(pb1:TPaintBox;p:TFigureInfo;Color:Tcolor); | pb1 | Объект, который сгенерировал событие |
| p |  |
| Color | Колонка |
| drawA | Обработка нажатие мыши | procedure drawA(pb1:TPaintBox;p:TArrowInfo; Color:TColor); | pb1 | Объект, который сгенерировал событие |
| p | кнопка |
| Color | Колонка |
| defaultDraw | Обработка клика | procedure defaultDraw(head:PFigList;pb1:TPaintBox); | head | Голова |
| DrawIF | Отрисовывает блок IF-Then-else | procedure DrawIF(pb1:TPaintBox; p:TFigureInfo; color:TColor); | pb1 | Область рисования |
| p |  |
| color | Колонка |
| DrawWhile | Рисует блок While | procedure DrawWhile(pb1:TPaintBox; p:TFigureInfo; color:TColor); | pb1 | Область рисования |
| DrawFigure |  | procedure DrawFigure(pb1:TPaintBox; p:TFigureInfo; color:TColor); | pb1 | Область рисования |
| p |  |
| DrawBlocks | Рекурсивная процедура, вызывающая отрисовку блоков и стрелок | procedure DrawBlocks(pb1:TPaintBox; head:PFigList; varmaxX,maxY:integer; ifst:TIfStates); | pb1 | Область рисования |
| head | Голова |
| axX |  |
| maxY | максимальный |
| ifst |  |
| DrawDirectArrows | Рисует стрелки для фигур фокуса ввода | procedure DrawDirectArrows(pb1:TPaintBox; p:TFigureInfo;left:boolean; ifState:TIfStates); | pb1 | Область рисования) |
| p |  |
| left |  |
| ifState | Статус |
| DrawArrow | Процедура отрисовывает стрелки | procedure DrawArrow(pb1:TPaintBox;inp,outp:TFigureInfo;isRight:Boolean;ifstate:TIfStates); | pb1 |  |
| inp | Исходящая фигура |
| :TFigureInfo;isRight |  |
| Boolean;ifstate | Статус |
|  |  |  | TIfStates |  |

Основное назначение модуля – отрисовывать фигуры и стрелки на объекте канвас компонента Paintbox.

## Описание модуля

## Описание модулей «Представления»

# ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Проведено тестирование программного средства. Тестирование программного средства производилась на персональном компьютере с установленной операционной системой Windows 10. Дополнительно работоспособность была проверена на виртуальной машине с установленной Windows XP.

## Тестирование функционала добавления фигур

Таблица 5.1 - тестирование функционала добавления фигур

## Вывод из прохождения тестирования

Программа успешно прошла все тесты, что показывает корректность работы программы и соответствие функциональным требованиям.

# РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

# ЗАКЛЮЧЕНИЕ

# СПИСОК ЛИТЕРАТУРЫ

[1] Глухова, Л.А. Основы алгоритмизации и программирования: учебное пособие. В 2 Ч. / Л.А. Глухова. – БГУИР, 2006 – Ч. 1. – 195 с.

[2] SVG – MDN web docs [Электронный ресурс] Режим доступа: https://developer.mozilla.org/docs/Web/SVG#Documentation — Дата доступа: 04.04.18

[3] Help for RAD Studio 10.2 Tokyo [Электронный ресурс] Режим доступа: http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Main\_Page— Дата доступа: 16.04.18

[4] А.Н.Вальвачев, К.А.Сурков, Д.А.Сурков, Ю.М.Четырько. Программирование на языке Delphi. Учебное пособие[Электронный ресурс] Режим доступа: http://www.rsdn.ru/?summary/3165.xml. Дата доступа: 20.04.18

# ПРИЛОЖЕНИЕ 1

**Схема алгоритма работы программы**

# ПРИЛОЖЕНИЕ 2

**Исходный код программы**

**Модуль главной формы:**