Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования (ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ПОСТРОЕНИЯ**

**ЛОГИЧЕСКИХ СХЕМ**

БГУИР КП I–40 01 01 421 ПЗ

Выполнил

студент: гр. 851004 Пашкевич А.Л.

Проверил: Варфоломеев А. В.

Минск 2019

Учреждение образования

«Белорусский государственный университет информатики и

радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2019г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Пашкевичу Антону Леонтьевичу* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы  *Программное средство для построения логических схем* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи законченной работы *31.12.2019г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Среда программирования C++ Builder. Разработать программное средство для построения логических схем. Реализовать набор инструментов, посредством которых пользователь осуществляет построение, включая расположение логических элементов на полотне, соединение элементов проводами, а также механизм ветвления проводов. Обеспечить возможность модификации схемы путём удаления элементов и проводов, перемещения их по полотну. Обеспечить корректность построение схемы, посредством валидации места расположения элементов и концов проводов. Реализовать режим моделирования, который предоставляет возможность проверить корректность функционирования построенной схемы, а также необходимые для него логические узлы. Реализовать экспорт созданного изображения в формат векторной графики SVG.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Руководство по установке и использованию программного средства\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Варфоломеев А.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *10.09.2019г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы):

*Раздел 1. Введение к 20.09.2019г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 10.10.2019г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 30.10.2019г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 30.11.2019г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Заключение. Приложения к 10.12.2019г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 20.12.2019г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 20.12.2019г. по 31.12.2019г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Варфоломеев А.В.*

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_\_\_Пашкевич А. Л. 10.09.2019г.*

*(дата и подпись студента)*

CОДЕРЖАНИЕ

[CОДЕРЖАНИЕ 4](#_Toc28199012)

[Введение 5](#_Toc28199013)

[1 Анализ предметной области 6](#_Toc28199014)

[1.1 Элементы теории логических схем 6](#_Toc28199015)

[1.2 Анализ существующих аналогов 8](#_Toc28199016)

[1.3 Представление векторного изображения в формате SVG 10](#_Toc28199017)

[2 Постановка задачи 12](#_Toc28199018)

[3 Разработка программного средства 13](#_Toc28199019)

[3.1 Информационная база задачи 13](#_Toc28199020)

[3.2 Схема алгоритмов решения задачи по ГОСТ 19.701-90 13](#_Toc28199021)

[3.2.1 Схема алгоритма AfterKeyWordsCheck 13](#_Toc28199022)

[3.4 Графический интерфейс 13](#_Toc28199023)

[4 Руководство по установке и использованию программного средства 14](#_Toc28199024)

[4.1 Работа с приложением 14](#_Toc28199025)

[Заключение 20](#_Toc28199026)

[Список использованной литературы 21](#_Toc28199027)

[Текстовые документы 22](#_Toc28199028)

[Пояснительная записка 22](#_Toc28199029)

[Графические документы 22](#_Toc28199030)

[Формат А1 22](#_Toc28199031)

Введение

Сегодняшний день просто немыслим без электронной вычислительной техники. Большинство таких приборов используют цифровую логику. При этом, сложные цифровые логические устройства, входящие в состав компьютера, состоят из ряда элементарных логических элементов, построенных на базе средств электронной техники.

Логические элементы позволяют реализовать любую логическую функцию. Входные и выходные сигналы логических элементов, соответствующие двум логическим состояниям 1 и 0, могут иметь один из двух установленных уровней электрического напряжения.

При проектировании цифровых логических устройств часто возникает задача по заданной таблице истинности записать выражение для логической функции и реализовать ее в виде логической схемы, состоящей из функционально полного набора логических элементов. Данная задача также называется задачей синтеза логических схем или логических устройств.

Синтез логических схем на основе функционально полного набора логических элементов состоит из представления логических функций, описывающих данные логические схемы в нормальных формах.

Данный курсовой проект направлен на создание эффективного приложения для построения логических схем, использующих различные функционально полные наборы логических элементов.

Актуальность такого продукта также подкрепляется широким распространением использования логических схем, а также небольшим выбором приложений, обеспечивающих простое, быстрое и наиболее комфортное построение данных схем.

В реализации проекта попутно решаются следующие задачи:

* разработка механизма моделирования процесса функционирования схемы;
* сохранение и открытие файлов, созданных с помощью данного приложения;
* предоставление возможности экспорта изображения в формат векторной графики SVG.

# Анализ предметной области

## Элементы теории логических схем

Логические схемы создаются для реализации в цифровых устройствах булевых функций (функций алгебры логики).

Булева функция в дискретной математике – это отображение Bn → B, где B = {0,1} — булево множество. Неотрицательное целое число n называют арностью или местностью функции, в случае n = 0 булева функция превращается в булеву константу. Каждая булева функция арности n полностью определяется заданием своих значений на своей области определения, то есть на всех булевых векторах длины n. Число таких векторов равно 2n. Поскольку на каждом векторе булева функция может принимать значение либо 0, либо 1, то количество всех n-арных булевых функций равно .

Булева функция задаётся конечным набором значений, что позволяет представить её в виде таблицы истинности, например как на рисунке 1.1.

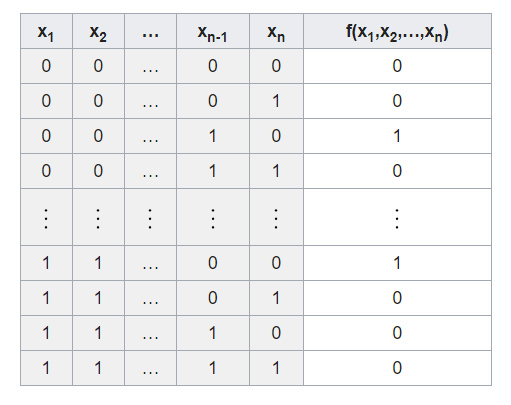


Рисунок 1.1 – Таблица истинности булевой функции

К основным булевым функциям относятся:

* отрицание;
* конъюнкция;
* дизъюнкция;
* сложение по модулю 2;
* импликация;
* эквивалентность;
* штрих Шеффера;
* стрелка Пирса.

Все сложные булевы функции возможно представлять таблицами истинности, но это не совсем удобно. Гораздо более простой способ их представления – это ДНФ и КНФ.

Дизъюнктивной нормальной формой или ДНФ называется дизъюнкция простых конъюнкций. При этом, простой конъюнкцией называется конъюнкция некоторого конечного набора переменных или их отрицаний, причём каждая переменная встречается не более одного раза. Например функция – ДНФ.

Совершенной дизъюнктивной нормальной формой или СДНФ относительно некоторого заданного конечного набора переменных называется такая ДНФ, у которой в каждую конъюнкцию входят все переменные данного набора, причём в одном и том же порядке.

Необходимо рассмотреть понятие функционального элемента. В общем случае, это устройство, предназначенное для обработки информации в цифровой форме. Функциональный элемент имеет входы и выходы. Сигналы на входах функционального элемента — аргументы функции, которую реализует функциональный элемент, сигналы на выходах — значение функции от аргументов. Если входные и выходные сигналы — являются нулями и единицами, элемент называется логическим. При подаче на входы логического элемента любой комбинации двоичных сигналов, на выходах также возникает сигнал — значение булевой функции.

На данном этапе стоит заметить, что в большинстве случаев логические схемы строятся именно по алгебраическому представлению булевой функции, которая прошла процесс минимизации и представлена в виде минимальной ДНФ. Методов минимизации булевых функции довольно много, но у всех них одна цель – сократить число логических операций (а это значит и логических элементов на схеме) до того минимума, при котором функция ещё не теряет своего исходного логического значения. Этот процесс, по сути своей, необходим в целях рационального использования материальных и технических ресурсов при производстве физической схемы, а также с целью уменьшения её размеров.

При построении логической схемы необходимо учитывать установленные в алгебре логики правила (приоритеты) для выполнения логических операций, которые в булевом базисе имеют следующую очерёдность: НЕ – И – ИЛИ.

При этом существует алгоритм построения логической схемы, на первом шаге которого определяется число логических переменных. Затем определяется количество логических операций и их порядок. На третьем шаге для каждой логической операции изображается соответствующий ей логический элемент. В конце логические элементы соединяются. Заметим, что соединение осуществляется в порядке выполнения операций. При этом, построение надо начинать с логической операции, которая должна выполняться последней.

## Анализ существующих аналогов

Как уже было замечено ранее, актуальность проектируемого продукта объясняется широким распространением использования логических схем, а также небольшим выбором приложений, обеспечивающих простое, быстрое и наиболее комфортное построение данных схем.

Среди ведущих аналогов выделяется среда моделирования Multisim от компании National Instruments. Данное программное обеспечение носит прикладной характер и служит для обучения схемотехнике, предназначается для курсов аналоговой, цифровой и силовой электроники, а также лабораторий.

Multisim включает в себя огромную стандартную библиотеку компонентов для эмулирования работы различных электронных схем. В Multisim встроен промышленный симулятор SPICE, а также MCU Module, позволяющий добавить в SPICE-эмулированную интегральную схему микроконтроллер и программировать его на С или Ассемблере. Модуль позволяет эмулировать работу интегральной схемы с микроконтроллером и различными дополнительными устройствами: RAM, ROM, клавиатурой, а также графическими и жидкокристаллическими дисплеями.

При этом построение конкретно логических схем – лишь верхушка айсберга возможностей данной среды моделирования. Вид схемы для функции , разработанной в Multisim представлен на рисунке 1.2.

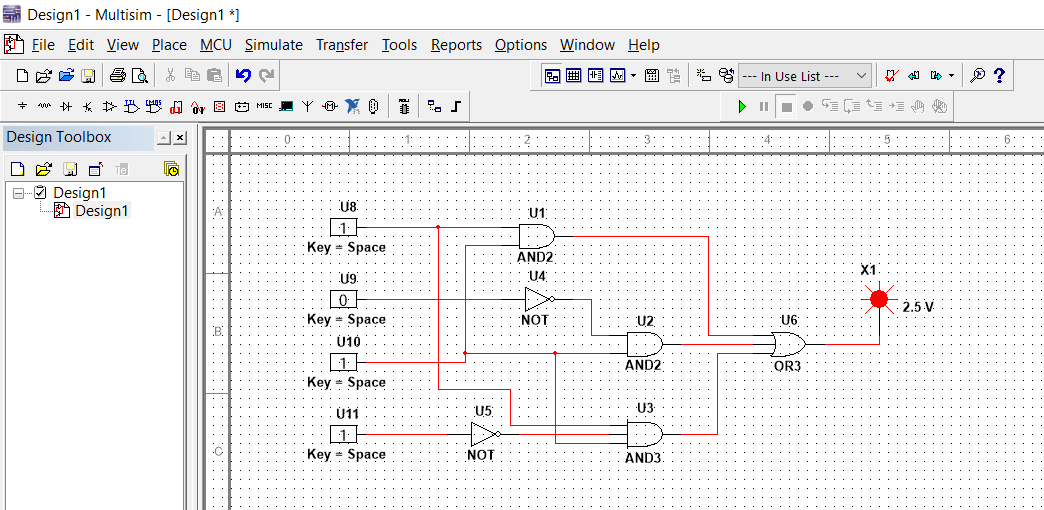


Рисунок 1.2 – Среда моделирования Multisim

К очевидным достоинствам приложения относятся широкие возможности моделирования и изучения работы электронных схем различного спектра назначения, а также опция программирования микроконтроллера на ассемблере. Однако помимо этого при анализе были выявлены и небольшие недостатки.

Во-первых, к ним относится избыточность приложения (если рассматривать его с целью построения конкретно логических схем булевых функций). Это значит, что Multisim настолько тяжеловесен, что для простой задачи создания небольшой схемы он не очень подходит. Во-вторых, с точки зрения пользователя процесс построения в некоторых местах неудобен из-за огрехов в режиме провода. Богатая библиотека также не способствуют быстрой адаптации работы с ней. В-третьих, в Multisim отсутствует экспорт изображения как в растровые, так и в векторные форматы. Наконец, Multisim не поддерживает логические элементы стандартов ГОСТ и МЭК, что является существенным недостатком для использования в странах СНГ.

Однако из данного приложения стоит позаимствовать такие элементы интерфейса схемы, как источник питания, лампочка, выделение компонента, подсветка входов компонента в режиме провода.

Ещё одним приложением такого же плана как Multisim является Proteus.

Proteus представляет собой систему схемотехнического моделирования, базирующуюся на основе моделей электронных компонентов, принятых в PSpice. Он предоставляет возможность моделирования работы программируемых устройств: микроконтроллеров, микропроцессоров, DSP. При этом в Proteus полностью реализована концепция сквозного проектирования, когда, например, инженер меняет что-то в логике работы схемотехники и программный пакет сразу же «подхватывает» данные изменения в системе трассировки. Библиотека компонентов содержит справочные данные. Дополнительно в Proteus входит система проектирования печатных плат. Proteus состоит из двух частей, двух подпрограмм: ISIS — программа синтеза и моделирования непосредственно электронных схем и ARES — программа разработки печатных плат.

Есть опция, позволяющая быстро написать программу для микроконтроллера, используемого в проекте, и скомпилировать.

Создадим схему функции и смоделируем её работу в приложении Proteus. Вид схемы представлен на рисунке 1.3.

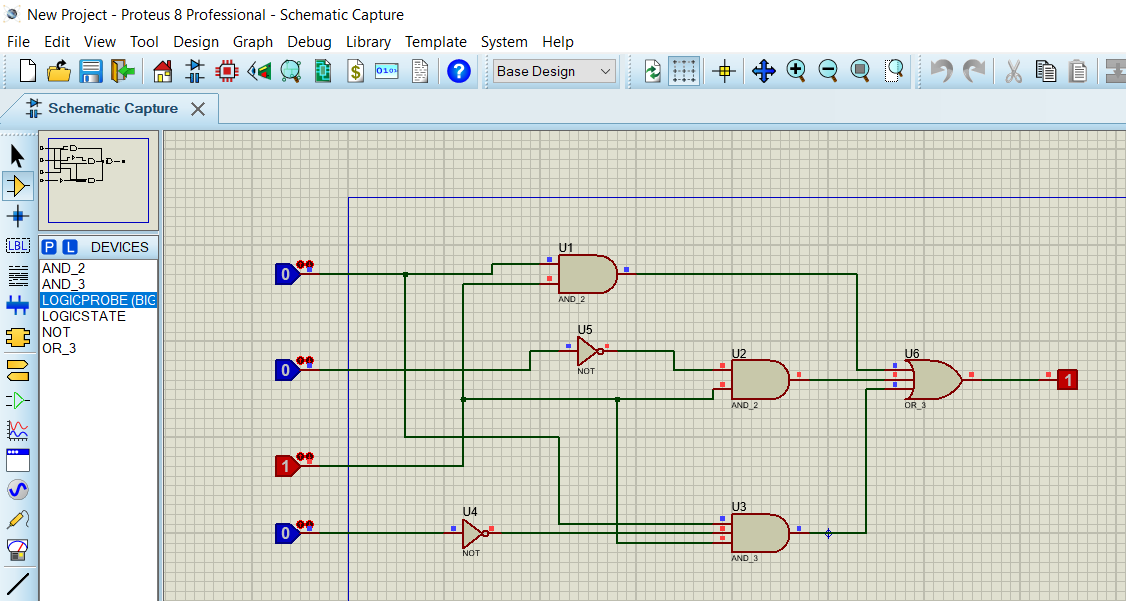


Рисунок 1.3 – Среда моделирования Proteus

В отличие от Multisim в Proteus присутствует возможность экспорта изображения в форматы BMP, PDF и SVG. Однако в остальном недостатки Multisim сохраняются. Приложение также не является простым в изучении и в процессе построения схемы. Поведение проводов не полностью контролируется пользователем.

При этом Proteus имеет довольно приятный режим моделирования, из которого стоит позаимствовать отображение сигналов на входах и выходах компонентов.

## Представление векторного изображения в формате SVG

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

Опишем структуру 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>

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" />

Для прямоугольника 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" />

Для окружности 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”/>

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" >LogicBuilder</text>

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

# Постановка задачи

Приложение LogicBuilder предназначается для построения логических схем линейных булевых функций и моделирования процесса их функционирования.

Таким образом, требуется разработать следующий функционал для построения логической схемы:

* структуру данных для хранения информации о компоненте;
* библиотеку компонентов стандарта ГОСТ;
* размещение компонентов на полотне;
* структуру данных для хранения информации о проводе;
* соединение компонентов проводами;
* перемещение компонентов по полотну;
* выделение и удаление компонентов и проводов;
* механизмы обновления информации о компонентах и проводах при удалении объекта;
* механизм валидации места нахождения компонента при его расположении и перемещении по полотну;
* корректное поведение подключенных к компоненту проводов при его перемещении;
* механизм валидации построения проводов посредством ограничения возможных стартовых и конечных точек проводов, а также выделением доступных точек;
* механизм ветвления проводов.

Для режима моделирования необходимо разработать:

* дополнительные компоненты «источник» и «лампа»;
* механизм распространения сигнала от источника по всей схеме;
* механизм вычисления логических значений на каждом компоненте, а также отображение входных и выходных сигналов. Для компонента -арной операции, где , упростить вычисление значения в случае лишь одного входного сигнала путём его пропуска и/или инвертирования;
* механизм определения присутствия в схеме циклов и, как следствие, неопределённого поведения.

Для работы приложения с файловой системой разработать:

* сохранение и открытие файлов с информацией о схеме;
* экспорт изображения схемы не в режиме моделирования в формат SVG.

# Разработка программного средства

## Информационная база задачи

## Схема алгоритмов решения задачи по ГОСТ 19.701-90

### Схема алгоритма AfterKeyWordsCheck

## Графический интерфейс

# Руководство по установке и использованию программного средства

## Работа с приложением

Для запуска приложения можно воспользоваться ярлыком или exe файлом, который находится в папке приложения. После появляется окно, изображенное на рисунке 5.6.

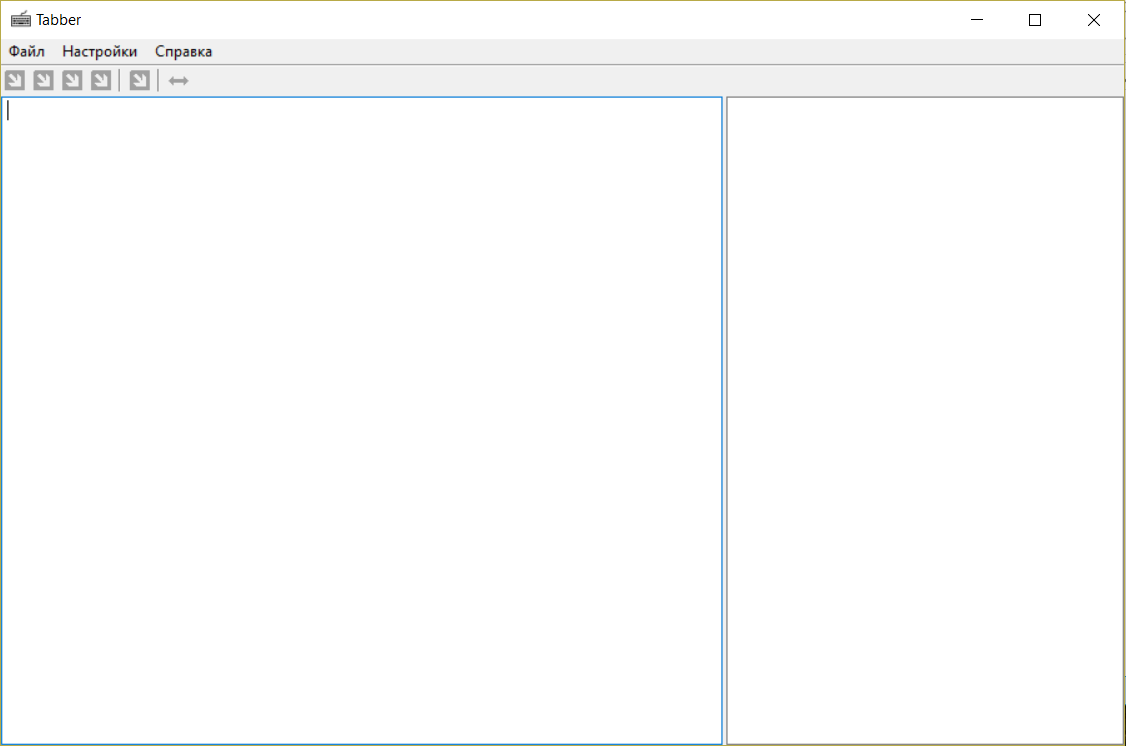


Рисунок 5.6 – Главное окно приложения

Для начала работы с Tabber необходимо открыть файл с текстом из вкладки главного меню «Файл», выбрав пункт «Открыть». Данная вкладка изображена на рисунке 5.7.

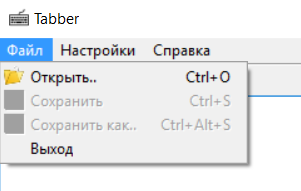


Рисунок 5.7 – Вкладка «Файл»

При нажатии появится диалог открытия файла, который предложит указать путь к необходимому файлу. После выбора пользователем директории, содержимое файла появится на первом текстовом поле.

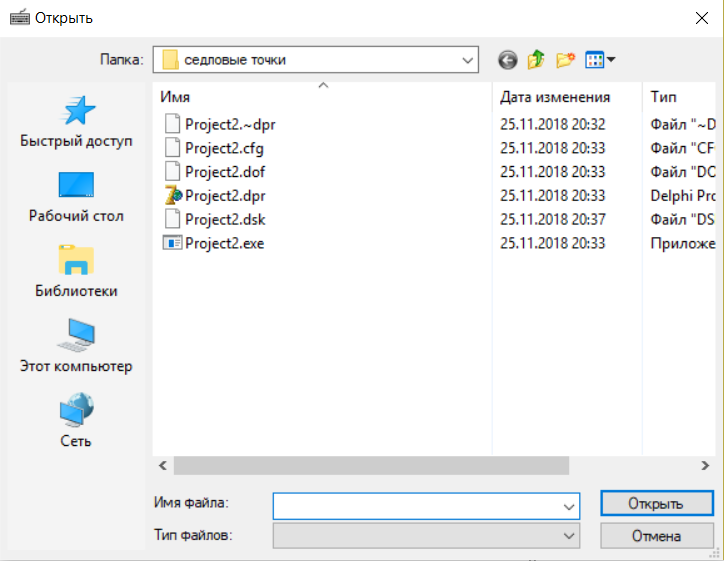


Рисунок 5.8 – Диалог открытия файла

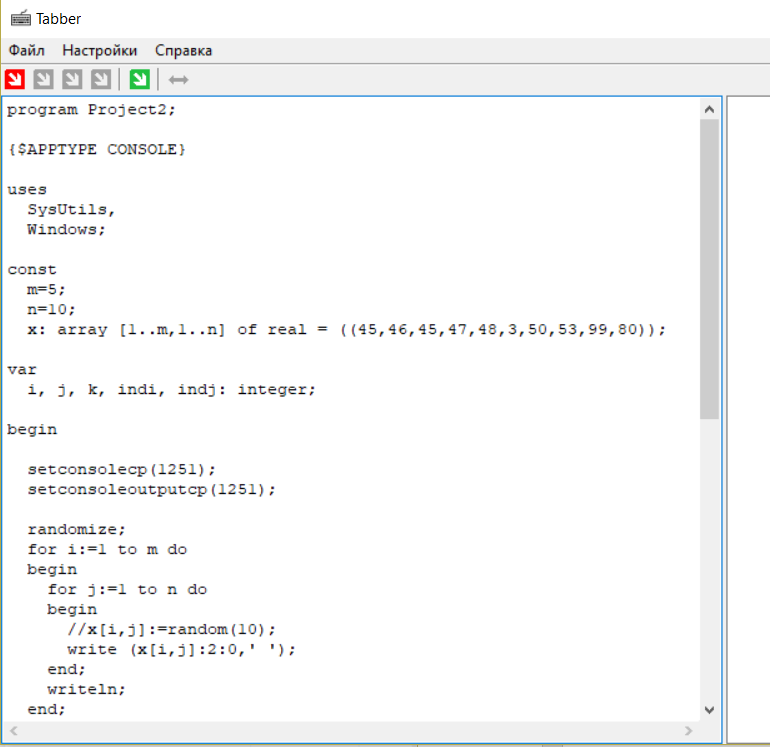


Рисунок 5.9 – Загрузка содержимого файла

Приложение поставляется без предустановленных настроек, поэтому при первом открытии пользователю необходимо настроить его самому. Для это надо открыть вкладку «Настройки» главного меню. Появится окно, изображённое на рисунке 5.10.

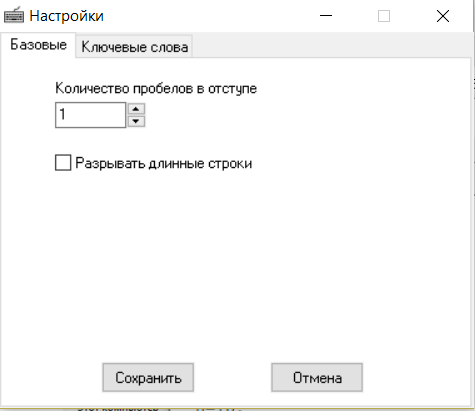


Рисунок 5.10 – Окно «Настройки»

Здесь необходимо указать количество пробелов в отступе, директиву приложению разрывать длинные строки и максимально возможную длину строки. На вкладке «Ключевые слова», которая изображена на рисунке 5.11, необходимо указать в каждом разделе зарезервированные слова языка Delphi, к которым будет чувствительно приложение.

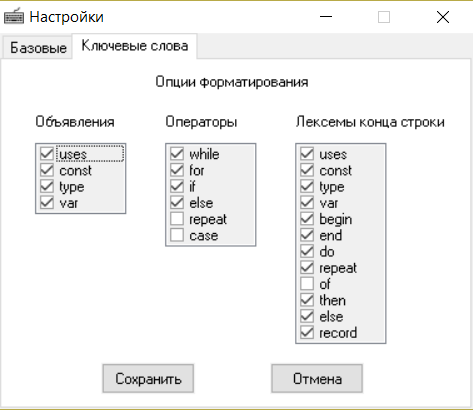


Рисунок 5.11 – Вкладка «Ключевые слова»

В разделе «Объявления» указываются ключевые слова объявлений, которые будут форматироваться. В разделе «Операторы» пользователь указывает те операторы, вложенные конструкции которых будут выделяться отступом. В разделе «Лексемы конца строки» указываются слова после которых будут разрываться строки, в случае, когда после них есть другие операторы.

После выбора опций, настройки необходимо сохранить нажатием на кнопку «Сохранить». Появится окно с рисунка 5.12.

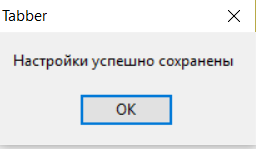


Рисунок 5.12 – Сохранение настроек

Сейчас приложение готово к работе. У пользователя есть панель инструментов обработки исходного текста.



Рисунок 5.13 – Панель инструментов

Пользователь может выбрать два пути: мгновенное форматирование нажатием зелёной стрелки и поэтапное форматирование. В обоих случаях результат обработки текста будет появляться на втором текстовом поле.

При поэтапном форматировании начальное нажатие происходит по красной стрелки, означающей сдвиг текста влево с разбиением строк по простым операторам. Затем становятся доступными остальные кнопки, которые можно нажимать в произвольном порядке, при этом они будут блокироваться после выполнения своей функции.



Рисунок 5.14 – Активная панель инструментов

Синяя стрелка отвечает за форматирование раздела операторов, оранжевая – за форматирование раздела объявлений, а фиолетовая разрывает длинные строки. Черная двунаправленная стрелка служит для переноса содержимого второго текстового поля на первое.

Результаты форматирования пользователь получает на втором текстовом поле.

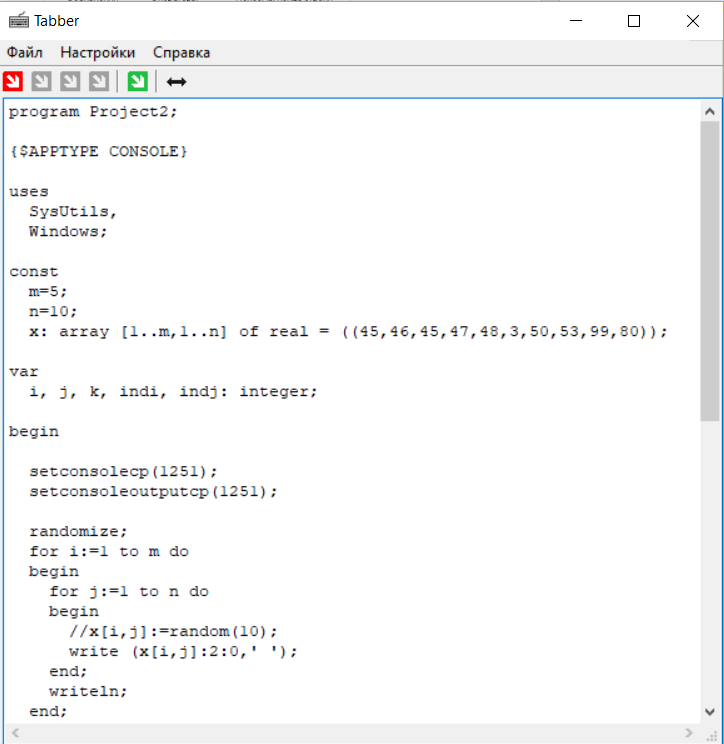


Рисунок 5.15 – Результаты работы приложения

По окончании работы изменения необходимо сохранить в уже открытый файл или же сохранить текст как новый файл. В случае выхода пользователя из приложения без сохранения изменений Tabber предложит эту возможность при закрытии – появится диалог с рисунка 5.16.

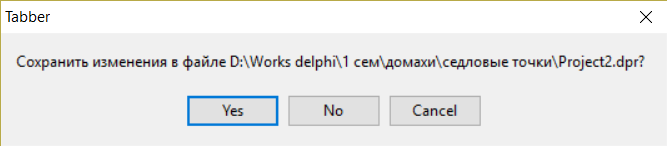


Рисунок 5.16 – Экстренное сохранение

После этого откроется диалог сохранения, в котором пользователь выбирает директорию сохранения файла.

При работе с приложением пользователь должен обратить внимание на то, что Tabber создаёт свои временные файлы в директории C:\Windows\TabberTemp\. По этому пути будут располагаться два файла: «input.txt» для хранения промежуточных результатов форматирования и «db.djdj», содержащий настройки пользователя. Категорически запрещается удалять данные файлы в процессе работы приложения.

Заключение

В ходе разработки было создано эффективное приложение для форматирования программного кода на языке Delphi. Помимо выполнения своей основной функции оно предоставляет пользователю широкие возможности по настройке процесса форматирования под себя. В качестве базового форматирования кода оно не уступает ведущим аналогам, но имеет большие недостатки в области детальной обработки текста.

Данное обстоятельство объясняется тем, что современные программы по форматированию кода, являются, по сути своей, полноценными компиляторами. Целью же данного курсового проекта было создание простого приложения для выполнения базовых функций, избавившись от сложного процесса синтаксического и лексического анализа текста. С этой задачей приложение справляется успешно.

Не потеряв своей основной концепции, данное приложение может быть улучшено посредством более точной обработки комментариев, внедрения анализа крупных ветвистых синтаксических конструкций и проверки корректности формирования управляющих операторов. Также пользователю могут быть предложены новые, более точные настройки форматирования.

В ходе разработки были усовершенствованы знания по динамическим структурам, работе с текстовыми и типизированными файлами, а также проектировании архитектуры крупного приложения. Приобретённые навыки использования графических компонентов среды разработки будут широко применяться в будущем.

Список использованной литературы

[1] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2007. – 4 ч.

[2] Серебряная, Л.В. Структуры и алгоритмы обработки данных: учеб. пособие / Л.В. Серебряная, И.М. Марина – Минск: БГУИР, 2013.

[3] Фленов, М.Е. Библия Delphi. – 3-e изд., перераб. И доп. / М.Е. Фленов –СПБ.: БХВ-Петербург, 2014. – 688 с.

[4] Пестриков, В.М. Delphi на примерах / В.М. Пестриков, А.Н. Маслобоев – СПБ.: БХВ-Петербург, 2005. – 496 с.

[5] Гофман, В.Э. Delphi. Быстрый старт. / В.Э. Гофман, А.Д. Хоменко –

СПБ.: БХВ-Петербург, 2003. – 288 с.

[6] Вирт, Н. Алгоритмы и структуры данных / Пер. с англ. Ткачев Ф.В. –

Москва: ДМК Пресс, 2010. – 272 с.

[7] Электронный ресурс: <https://edn.embarcadero.com/article/10280>.

[8] Пратт, Т. Языки программирования: разработка и реализация — 4-е издание. — Питер, 2002. — 688 с.

ВЕДОМОСТЬ ДОКУМЕНТОВ

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | | *Дополнительные сведения* | |
|  | | | | | Текстовые документы | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР КП 1–40 01 01 423 ПЗ | | | | | Пояснительная записка | | | | 118 с. | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | Графические документы | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.851004-01 СА | | | | | Форматирование раздела объявлений программного кода на языке Delphi | | | | Формат А1 | |
|  | | | | | Схема алгоритма | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  | БГУИР КП I- 40 01 01 423 ПЗ | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм | Лист | № докум. | Подп. | Дата | Программное средство для форматирования программного кода на языке Delphi  Ведомость курсового  проекта | Литера | | | Лист | Листов |
| Разраб. | | Пашкевич А.Л. |  | 27.05 | Т |  |  | 1189 | 118 |
| Провер. | | Фадеева Е.П. |  | 31.05 | Кафедра ПОИТ  гр. 851004 | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |