Министерство народного образования

**Приднестровский Государственный Университет**

**им. Т.Г. Шевченко**

## Физико-математический факультет

**Кафедра общей физики и методики преподавания физики**

**Кафедра информатики и вычислительной техники**

### Дипломная работа

Математическое моделирование физических задач на ЭВМ

На примере расчета разветвленных цепей постоянного

тока основанного на использовании законов Кирхгофа

Выполнил:

Студент 506 группы

Коваленко А.С.

Научные руководители:

Кандидат

физико-математических

наук, доцент Цыпишка Д.И.

Кандидат

физико-математических

наук, доцент Брагарь Л.Ф.

Тирасполь, 2000г.

Содержание:

Введение 3

1. Напряжение и ток в электрической цепи 4

2. Резистивный элемент 5

3. Источники 6

Глава 1. Задача анализа разветвленной цепи 7

Глава 2. Пример. Результаты вычислений 9

Глава 3. Методика моделирования 10

1. Линейный граф и матрица соединений 10

2. Уравнения контурных токов 13

3. Алгоритм формирования узловых уравнений 16

Заключение 17

Использованная литература 18

Приложение 19

# 

# Введение

Все электротехнические и радиотехнические устройства пред­ставляют собой электромагнитные устройства, главные про­цессы в которых подчиняются общим законам электромагне­тизма. В любом электромагнитном устройстве происходит движение электрических зарядов, неразрывно связанное с изменяющимся во времени и пространстве электромагнитным полем, двумя сторонами которого являются электрическое и магнитное поля.

Электромагнитные процессы сопровождаются взаимным преобразованием электромагнитной энергии в другие виды энергии. Точный анализ этих процессов, описываемых систе­мами уравнений в частных производных (уравнениями Макс­велла), - задача, трудно разрешимая даже в простейших слу­чаях. Но для инженерных расчетов и проектирования устройств необходим количественный анализ. Поэтому возникает потреб­ность в приближенных методах анализа, позволяющих с достаточной степенью точности решать широкий круг задач. Такие методы дает теория электрических цепей, которая для характеристики электромагнитных процессов вместо векторных величин теории поля, зависящих от пространственных коорди­нат и времени, вводит интегральные скалярные величины – ток и напряжение, являющиеся функциями времени.

Для приближенного учета процессов преобразования электромагнитной энергии в теории цепей вводят идеальные элементы с выводами или полюсами, через которые проходит электрический ток. Простейшими идеальными, базисными эле­ментами являются двухполюсные элементы с двумя полюсами или выводами – индуктивный, емкостный и резистивный эле­менты, учитывающие накопление энергии в магнитном и электрическом полях и необратимое преобразование электро­магнитной энергии в другие виды энергии. Для учета преобра­зования энергии неэлектрической природы (химической, меха­нической, тепловой и т. д.) в электромагнитную энергию вводят элемент с двумя выводами, называемый источником. Наряду с указанными вводят четырехполюсные и многополюсные эле­менты в общем случае с *n* выводами.

Соединяя между собой соответствующим образом эти идеальные элементы, получают электрическую цепь, приближенно отображающую электромагнитные процессы в каком-либо устройстве по отношению к интересующим выводам.

Теория цепей применима к большому числу устройств, в которых представляют интерес процессы в отдельных точках – выводах.

В настоящее время существуют методы и средства расчета радиотехнических цепей на основе математических моделей, представляющие собой в общем случае системы нелинейных дифференциальных уравнений. Одним из многих таких средств является программа, предложенная в [1], которая представляет собой реализацию математической модели расчета цепей постоянного тока. Программа работает следующим образом: пользователь вводит все данные для расчета цепи, самостоятельно производя анализ цепи, т.е. он вводит количество узлов, количество ветвей с элементами, находящимися на них и номиналы этих элементов. Програма решает получающиеся при этом линейные уравнения и выводит результат вычислений.

Недостатком указанных выше программных средств является отсутствие автоматизированного построения разветвленных цепей, ввода элементов, выбора направления обхода контуров и токов в ветвях по введенной принципиальной схеме. Кроме этого существующие программы не позволяют непосредственно при расчетах проводить анализ полученных результатов, в динамике изменять параметры компонентов.

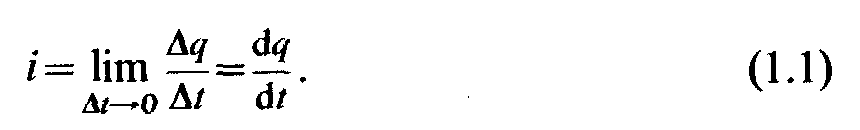
В связи с этим целью дипломной работы является: разработка математической модели и программы анализа и расчета цепей постоянного тока, автоматического выбора направления обхода контура и направления токов в ветвях цепи, и выводить результаты вычислений.

В данной дипломной работе рассматривается метод расчета и анализа линейных разветвленных цепей содержащих резистивные элементы и источники ЭДС с постоянными пара­метрами элементов основанный на использовании законов Кирхгофа.

## 1. Напряжение и ток в электрической цепи

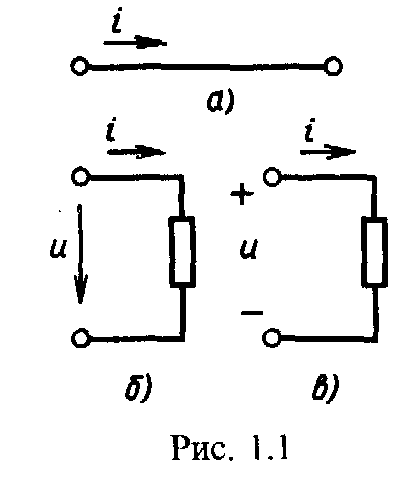
Электрический ток и напряжение являются основны­ми величинами, характеризующими состояние электрических цепей.

Электрический ток в проводниках представляет явление упорядоченного движения электрических зарядов. Под терми­ном «ток» понимают также интенсивность или силу тока, измеряемую количеством электрического заряда *q,* прошед­шего через поперечное сечение проводника в единицу вре­мени:



Следовательно, *ток представляет собой скорость изменения заряда во времени.* В СИ заряд выражается в кулонах (Кл), время – в секундах (с), ток – в амперах (А).

Ток как отношение двух скалярных величин является скалярной алгебраической величиной, знак которой зависит от направления движения зарядов одного знака, а именно условно принятого положительного заряда. Для однозначного опреде­ления знака тока за положительное направление достаточно произвольно выбрать одно из двух возможных направлений, которое отмечают стрелкой (рис. 1.1, а).

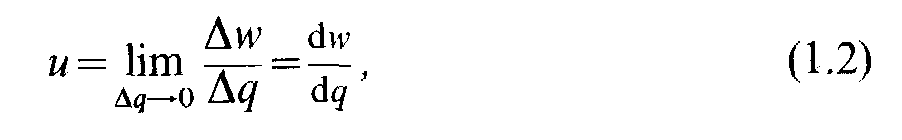


Если движение поло­жительного заряда происходит в направлении стрелки, а движение отрицательного заряда—навстречу ей, то ток поло­жителен. При изменении направления движения зарядов на противоположный ток будет отрицательным.

Перед началом анализа на всех участках цепи необходимо отметить положительные направления то­ков, выбор которых может быть произ­вольным. Программа расчета построена так, что за положительное направление тока принято направление движения «по часовой стрелке».

Прохождение электрического тока или перенос зарядов в цепи связаны с преобра­зованием или потреблением энергии. Для определения энергии, затрачиваемой на перемещение заряда между двумя рассмат­риваемыми точками проводника, вводят новую величину – напряжение.

Напряжением называют количество энергии, затрачи­ваемой на перемещение единицы заряда из одной точки в другую:



где *w—*энергия.

При измерении энергии в джоулях (Дж) и заряда в кулонах (Кл) напряжение выражают в вольтах (В).

Напряжение как отношение двух скалярных величин также является скалярной алгебраической величиной. Для однознач­ного определения знака напряжения между двумя выводами рассматриваемого участка цепи одному из выводов условно приписывают положительную полярность, которую отмечают либо стрелкой, направленной от вывода, либо знаками «+», «-» (рис. 1.1, б, в). Напряжение положительно, если его поляр­ность совпадает с выбранной; это означает, что потенциал вывода со знаком «+», из которого выходит стрелка, выше потенциала второго вывода.

Перед началом анализа должны быть указаны выбранные положительные полярности напряжений – только при этом условии возможно однозначное определение напряжений. В программе по умолчанию каждому источнику ЭДС приписывают «+» к высшему потенциалу, а «-» – к низшему.

Положительную полярность напряжения выбирают согласованной с выбранным положительным направлением тока, когда стрелки для тока и напряжения совпадают или знак «+» полярности напряжения находится в хвосте стрелки, обозначающей положительное направление тока. При согласо­ванном выборе полярности, очевидно, достаточно ограничиться указанием только одной стрелки положительного направления тока.

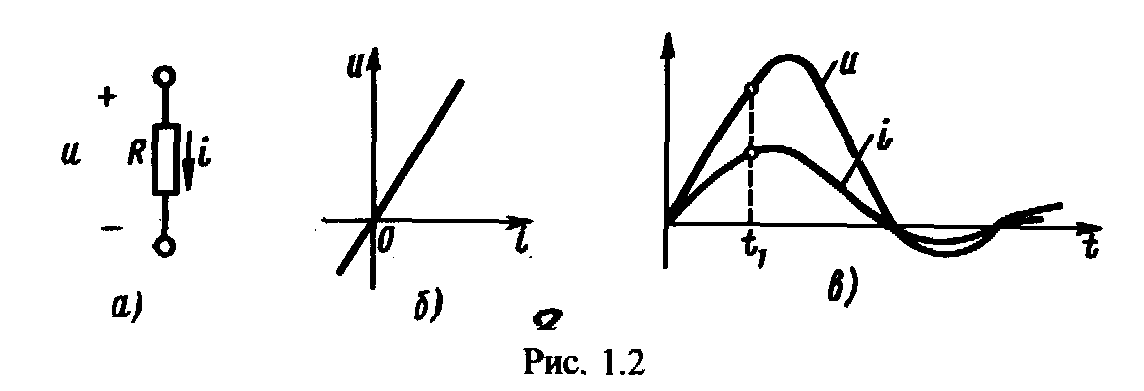
Для обозначения условно положитель­ной полярности применяют знаки «+», «-» у выводов участка цепи.

## 2. Резистивный элемент

Под резистивным элементом электрической цепи или активным сопротивлением понимают идеализированный эле­мент, в котором происходит только *необратимое преобразование электромагнитной энергии в теплоту или другие виды энергии,* а запасание энергии в электрическом и магнитном полях от­сутствует.

По свойствам к этому идеальному элементу довольно близки такие реальные устройства, как угольные сопротивления, реостаты, лампы накаливания при относительно небыстрых изменениях токов.

Условное графическое обозначение резистивного элемента Представлено на рис. 1.2, *а,* где указаны принятые положитель­ные направления напряжения и тока.



Основное уравнение элемента, связывающее ток и напряжение, так называемая вольт-амперная характеристика, определяется законом Ома, который устанавливает пропорциональ­ность между напряжением и током:

U=RI, I=GU (1.3)

Коэффициент пропорциональности в первом выражении (1.3), равный отношению напряжения и тока, является электрическим сопротивлением:

R=U/I (1.4)

Численно сопротивление равно напряжению на элементе при токе в 1 А. Значение сопротивления выражается в омах.

Обратная величина – отношение тока к напряжению – представляет собой электрическую проводимость:

G=I/U=1/R. (1.5)

В теории линейных электрических цепей сопротивление и проводимость принимают постоянными, не зависящими от тока, напряжения и других величин. В реальных элементах это допущение, так же как и допущение отсутствия запасания энергии, выполняется приближенно.

## 3. Источники

Под источником в теории цепей понимают элемент, питающий цепь электромагнитной энергией. Эта энергия по­требляется пассивными элементами цепи – запасается в индуктивностях и емкостях и расходуется в активном сопротивлении.

Напряжения источников, представляющие задан­ные функции времени, называют также приложенными к цепи или возбуждающими цепь сигналами. Примерами реальных источников электромагнитной энергии могут служить генераторы постоянных, синусоидальных и импульсных сигналов разнообразной формы, сигналы, полу­чаемые от различного рода датчиков, антенн радиоприемных устройств и т. д. Эти источники сигналов либо являются первичными источниками, в которых происходит непосредст­венное преобразование энергии неэлектромагнитной природы (механической, химической, тепловой и т. д.) в электромагнит­ную энергию, либо получают питание от первичных источников. Источник является актив­ным элементом.

Для анализа цепей вводят идеализированный источник напряжения, который учитывает главные свойства реального источника.

***Источник напряжения.*** Под источником напряжения понимают такой элемент с двумя выводами (полюсами), напряжение между которыми задано в виде некоторой функции времени независимо от тока, отдаваемого во внешнюю цепь.

а) б)

Рисунок 1.3.

Наиболее часто применяемые условные графические обо­значения источника напряжения представлены на рис. 1.3, а и б, где принятая положительная полярность напряжения источника указывается либо стрелкой внутри кружочка, либо большой и малой чертами, малая соответствует знаку «-», а большая - «+». Поскольку положительную полярность напряжения усло­вились обозначать знаками «+», «-», для источника напряже­ния в программе применено обозначение, показанное на рис. 1.3, б.

# Глава 1. Задача анализа разветвленной цепи

Электрическую цепь, приближенно отображающую электро­магнитные процессы в реальном устройстве, составляют путем соответствующего соединения между собой рассмотренных двухполюсных элементов: сопротивления, индуктивности, ем­кости и источников сигнала. В общем случае отдельные элементы, а также отдельные участки цепи могут соединиться произвольно. В дипломной работе рассмотрены только соединение сопротивления и источника сигнала, в качестве которого используют ЭДС.

В результате получается электрическая схема, имеющая определенную геометрическую конфигурацию. На рис. 4 приложения показан пример схемы электрической цепи, составлен­ной из нескольких сопротивлений и источников ЭДС.

Основными понятиями, характеризующими геометрическую конфигурацию разветвленной цепи, являются ветвь и узел.

Под ветвью в общем случае понимают участок цепи с двумя выводами. Токи ветви принимают в качестве неизвестных переменных, характеризующих состояние цепи. Поэтому, что конкретно следует понимать под ветвью, зависит от выбора переменных цепи. Ветвью можно считать каждый элемент цепи. Но для уменьшения числа переменных за ветви иногда принимают также участки из последовательного соеди­нения отдельных элементов, токи которых имеют одно и то же значение, и участки из параллельного соединения отдельных элементов, напряжения на которых имеют одно и то же значение. При анализе схемы за ветвь принимается участок цепи между двумя узлами цепи.

Узел электрической цепи – это точка на схеме, в которой сходятся более двух ветвей [4]. Например, на рисунке №4 приложения – 4 узла.

*Задача анализа электри­ческой цепи*формулируется та­ким образом: Заданы схема электрической цепи со значения­ми всех ее элементов, а также напряжения источников, действующих в цепи. Требуется найти токи ветвей. В дальнейшем будем применять общие термины, назы­вая заданные напряжения источников функциями воз­буждения или сигналами, а искомые токи вет­вей, определяемые в результате анализа цепи, - реакциями. Следовательно, требуется найти реакции цепи на действие заданных сигналов.

Выводы – узлы или ветви, реакции которых необходимо найти, - называют выходными, а выводы, к которым при­соединены источники, - входными.

Программа предназначена для анализа любой линейной цепи произвольной конфигура­ции с любым конечным числом элементов.

Для определения искомых реакций – токов ветвей в общем случае – необходимо со­ставить уравнения цепи с помощью двух систем уравнений:

1) уравнений элементов, связывающих ток и напряже­ние каждого элемента, а также заданные напряжения. Уравнения элементов не зависят от схемы и геометрической конфигурации цепи, в которую входят элементы;

2) уравнений соединений, которые определяются только геометрической конфигурацией и способами соединений ветвей (элементов цепи) и не зависят от вида и характера элементов. Уравнения соединений устанавливают связи между токами и напряжениями отдельных элементов, входящих в цепь.

Уравнения соединений составляют па основе двух законов Кирхгофа, которые связывают токи ветвей, сходящихся в узлах, и напряжения ветвей, входящих в контуры; контуры представляют замкнутые пути, проходящие однократно через ряд ветвей и узлов.

**Первый закон Кирхгофа**, выражающий закон сохра­нения заряда, дает уравнение равновесия токов в узле цепи и формулируется так: *в любой момент алгебраическая сумма токов ветвей, сходящихся в узле электрической цепи, равна нулю:*

.

Знак тока определяется выбором положительных направле­ний токов ветвей; токам, выходящим из узла, приписывают условно знак «-», а током, входящим в узел, - знак «+».

**Второй закон Кирхгофа**, выражающий закон сохра­нения энергии, дает уравнение равновесия напряжений в контуре и формулируется следующим образом: *в любой момент алгебраическая сумма напряжений ветвей в контуре равна нулю*

.

Знак напряжения определяется выбором положительных полярностей напряжений ветвей: если при обходе контура перемещение происходит в сторону понижения или падения напряжения, то напряжению ветви условно приписывают знак «+», если в сторону повышения напряжения - знак «-».

Линейные цепи, составленные из элементов одного вида, например резистивных, описываются системами линейных ал­гебраических уравнений.

Применяя программу расчета линейных разветвленных электрических схем, необходимо лишь нарисовать схему, и ввести все значения сопротивлений и ЭДС. Все остальные преобразования, такие как выбор обхода контура, направления ЭДС, программа выполнит сама и выдаст конечный результат – значения токов в ветвях схемы.

Целью настоящей дипломной работы является создание математической модели и программы работающей по этой модели, позволяющей анализировать и расчитывать разветвленные электрические цепи постоянного тока, на основе использования законов Кирхгофа.

На основе проведенного литературного обзора я убедился, что в настоящее время существуют только программы, которые решают лишь уравнения созданные при анализе цепи, но не производят анализ самой цепи.

# Глава 2. Пример. Результаты вычислений

**Задача** [3, №1.50]

Дано:

Е1=120В; Е2=60В; Е3=140В;

R1=1Ом; R2=0,5Ом; R3=0,4Ом; R4=R5=R6=3Ом

Найти токи в ветвях.

Ответ задачи: I1=6,8; I2=30,9; I3=24,1; I4=12,6; I5=18,3; I6=5,8.

E1 R1

E2 R2 R4

R6

R5

E3 R3

Схема для задачи:

E1 R1

E2 R2 R4

R6

R5

E3 R3

Эквивалентная схема для программы:

Результат вычисления программы:

Ответ: I1=6,83; I2=30,88; I3=24,05; I4=12,57; I5=18,31; I6=5,74.

Как видно, программа дает более точный результат, чем тот, который предлагается для проверки правильности решения задачи.

Результаты вычислений выводятся в отдельном окне. (Рисунок №6 приложения).

# Глава 3. Методика моделирования

В этой главе излагаются общие методы анализа цепей произвольной структуры, составленных из двух­полюсных резистивных элементов с постоянными сопротивле­ниями и ЭДС, использованные для анализа схем в программе. Методы основаны на составлении уравне­ний цепи относительно выбранных переменных и их решении.

## 1. Линейный граф и матрица соединений

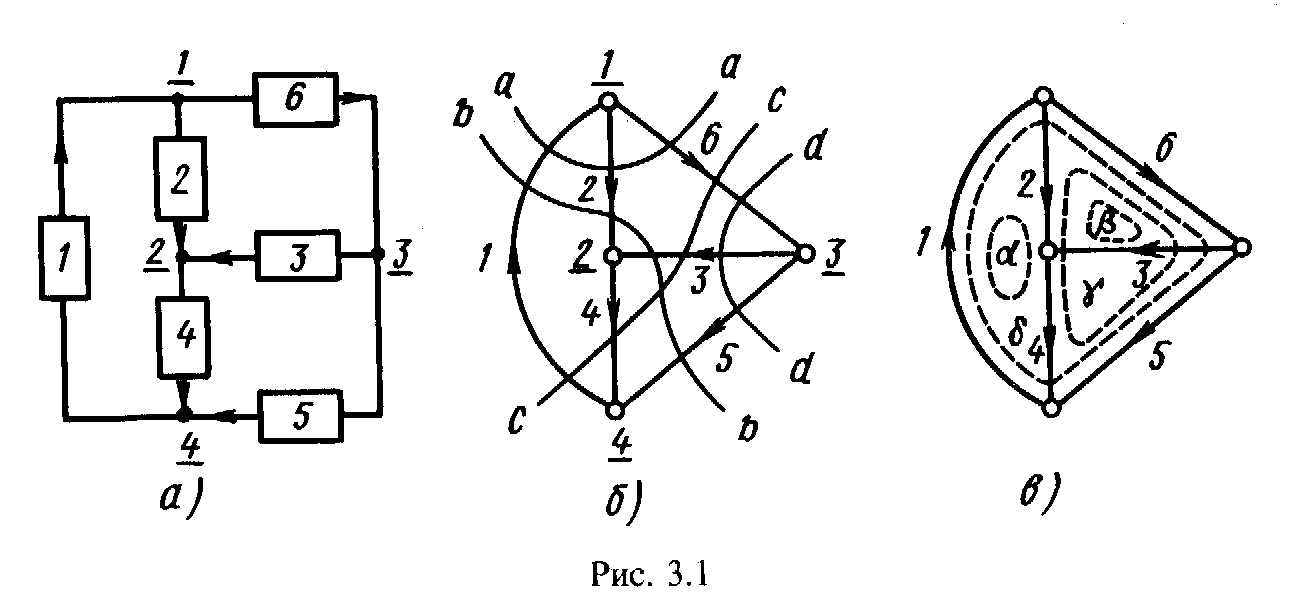
Для цепей сложной структуры использовалась за­пись уравнений в матричной форме. Матричная запись:

1. позволяет распространять формальным образом полученные уравнения на цепи любой сложной структуры;
2. системати­зирует и упрощает процесс составления уравнений;
3. дает алгоритмы формирования уравнений с помощью ЭВМ; в случае сложных цепей составление уравнений «вручную» (без ЭВМ) требует значительных затрат времени.

Рассмотрим классические методы контурных и узловых уравнений. Вначале введем понятие графа цепи, описы­вающего свойства цепи, связанные с взаимным соединением ветвей, т. е. с геометрической структурой (топологией) схемы. Применение понятия графа позволяет записывать в матричной форме уравнения соединений, составляемые на основе законов Кирхгофа, и тем самым формировать уравнения разветвленных цепей с помощью ЭВМ.

Уравнения равновесия токов и напряжений, составленные по Законам Кирхгофа, как указывалось, линейными однородными уравнениями. Важное условие, которое должно обеспечиваться, состоит в линейной независимости уравнений. Ни одно уравнение не должно быть получено линейной комбинацией остальных уравнений. Общий систематический метод получения линейно независимых уравнений цепи основан также на привлечении понятий теории линейного графа, одного разделов математической дисциплины—топологии. К линейному графу приводит следующее соображение:

Уравнения равновесия токов и напряжений, составленные по законам Кирхгофа, определяются только схемами соединений ветвей, т. е. геометрической структурой цепи, и не зависят от вида и характеристик элементов, т. е. от физического со­держания ветвей. Поэтому при составлении уравнений со­единений удобно отвлекаться от вида и характеристик ветвей цепи, заменив их линиями. В результате для цепи рис. 3.1, а, составленной из любых двухполюсных элементов, получим линейный граф, показанный на рис. 3.1, б.



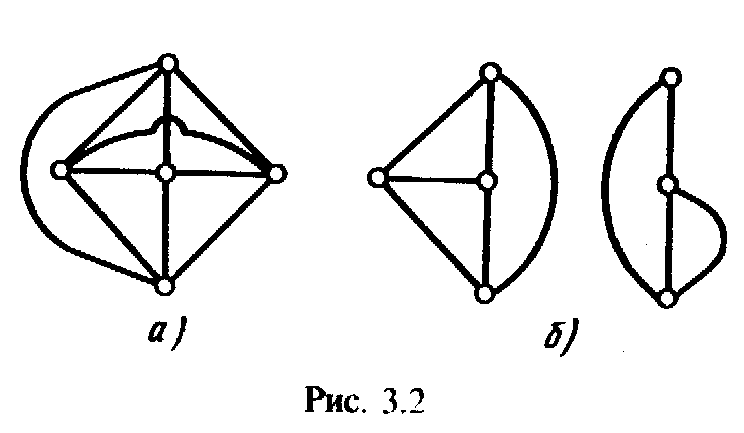
*Граф является системой или совокупностью двух элемен­тов—узлов (вершин), изображаемых точками, и ветвей (ре­бер), изображаемых отрезками линий, которые соединяют пары узлов.* В предельном вырожденном случае граф может состоять только из одного узла.

Числа узлов и ветвей графа обозначим *пy* и *nд*. Поскольку каждому узлу и каждой ветви цепи сопоставляется узел и ветвь графа, граф цепи содержит всю информацию о соединениях и геометрических свойствах исходной цепи. На рис. 3.1, а, *б* соответственные узлы, а также ветви цепи и графа имеют одинаковые номера.

Граф, так же как и исходная цепь, может иметь различную структуру. Различают планарный (плоский) граф, если его можно изобразить на плоскости без пересечения ветвей (рис. 3.1,6), и не планарный (пространственный) граф, если при его изображении на плоском чертеже невозможно избежать пересечения ветвей (рис. 3.2, а). *Полным* назы­вают граф, у которого каждая пара узлов соединена одной ветвью. Примером полного графа цепи может служить граф рис. 3.2, а.

Любую часть графа, элементы которой являются элемен­тами исходного графа, называют подграфом. Подграф получают путем удаления (исключения) некоторых ветвей исходного графа.

Важным подграфом является путь графа, представляю­щий непрерывную последовательность ветвей, связывающую пару выбранных узлов, с прохождением каждого узла не болееодного раза. Смежные вет­ви пути имеют общий узел, так что к каждо­му узлу присоединены две ветви, лишь к край­ним узлам — по одной ветви.

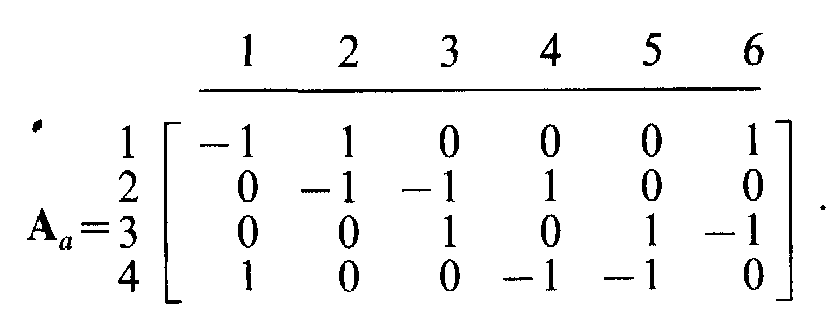


На рис. 3.1, б пути, свя­зывающие узлы 1, 4*,* образованы ветвями 2-4*, 5-6, 1, 2-3-5* и т. д. Если в заданном графе имеется хотя бы один путь между любой парой узлов, то граф называется связным—он соответствует цепи, элементы которой соединены только электрически. Граф рис. 3.1, б является примером связного графа, а рис. 3.2, б —несвяз­ного: он состоит из двух раздельных частей, элементы ко­торых могут иметь связь, например, через взаимную ин­дуктивность.

Для составления уравнений соединений по законам Кирх­гофа необходимо на всех ветвях графа стрелками указать положительные направления токов. В результате получается граф с ориентированными ветвями, называемый направлен­ным графом токов цепи (рис. 3.1, б), ветви которого явля­ются токами. Положительные полярности напряжений ветвей удобно принимать согласованными с положительными на­правлениями токов. Тогда в цепях, составленных из двух­полюсных элементов, направленный граф напряжений, реб­ра которого являются напряжениями ветвей, будет совпа­дать с графом токов. Переход к направленному графу позволяет производить аналитическую запись структуры графа и подграфов в виде таблиц – матриц, называемых топологическими матрицами. Аналитическое представ­ление графа необходимо для формирования уравнений сложной цепи с помощью ЭВМ.

Полное описание структуры направленного графа дает *nу*x*nв -* матрица соединений, n*у* строк ко­торой являются порядковыми номерами узлов, nв столб­цов – номерами ветвей. Элементами *аi,j* этой матрицы яв­ляются символы наличия или отсутствия ветви *k,* присое­диненной к узлу *i,* которые принимаются равными +1 (—1) для выходящей из узла (входящей) ветви и 0, если ветвь не связана с узлом.

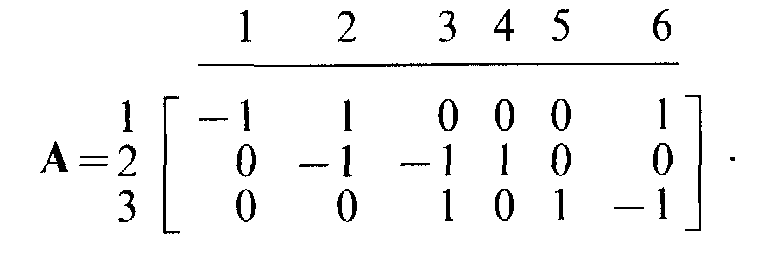
Для того чтобы записать матрицу соединений, достаточно для каждой ветви определить номера обоих соединяемых узлов i, *j* и заполнить клеточки на пересечениях строк *i, j* и столбца с номером ветви *k* значениями +1, — 1; в остальных клеточках должны быть проставлены нули. Для графа рис. 3.1,б получим полную матрицу соединений:



(3.1)

Так как каждая ветвь соединяет два узла—выходит из одного узла и входит в другой, то столбец матрицы состоит из двух ненулевых элементов +1, —1 (их сумма равна нулю), так что достаточно заполнить таблицу для ny*-1* узлов, которая является редуцированной матрицей соединений А. Эту незави­симую матрицу можно получить из полной матрицы Аa вычеркиванием строки, соответствующей выбранному базисно­му узлу.

Приняв в качестве базисного узел *4* и соответственно вычеркивая четвертую строку в (3.1), получим редуцированную матрицу соединений:



(3.2)

Строка матрицы А показывает, какие ветви выходят из каждого независимого узла графа цепи (и входят в него), а столбец – к каким узлам присоединена ветвь.

В отличие от полной матрицы Аа у реду­цированной матрицы соединений связного графа множест­во всех строк линейно независимо. Отсюда можно сделать вывод о том, что система уравнений равновесия токов в ny-1 узлах цепи линейно независима. Если ввести вектор токов *пв,* ветвей:

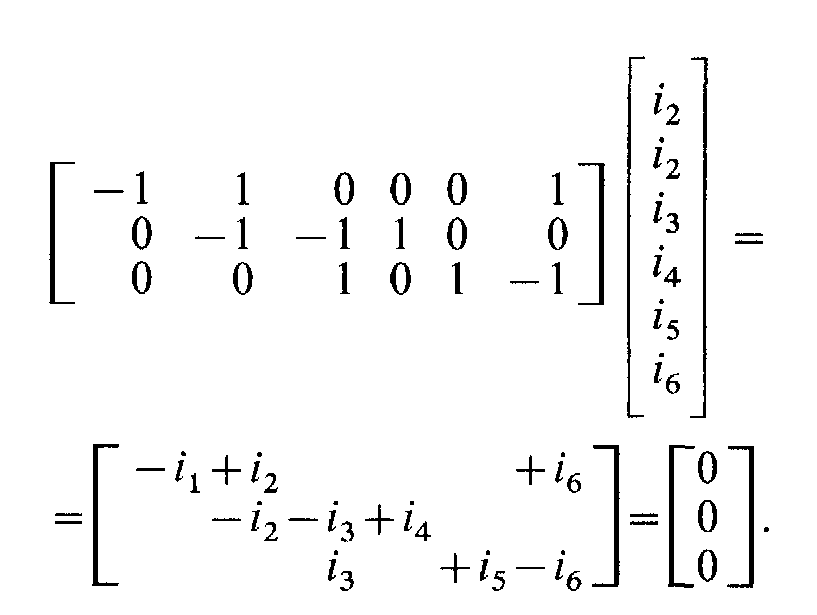
i=[i1, i2, … inв]Т, (3.3)

то систему независимых уравнений в n*у-*1 узлах по **ЗТК** в соответствии со смыслом матрицы А можно записать в виде:

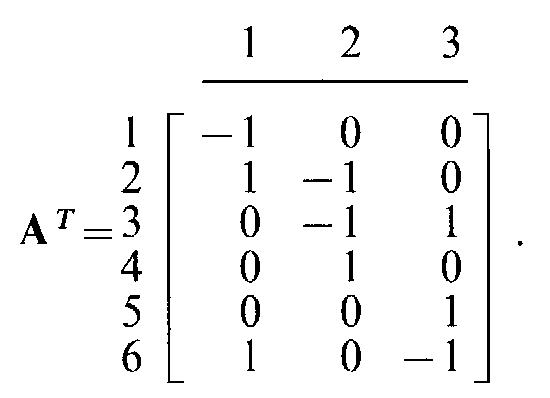
Ai=0, (3.4)

где 0=[0 0 … 0]T - нулевой вектор размерности n*у-*1.

Для графа цепи рис. 3.1, б с матрицей соединений (3.2) имеем:



Транспонированная матрица соединений имеет вид:



(3.5)

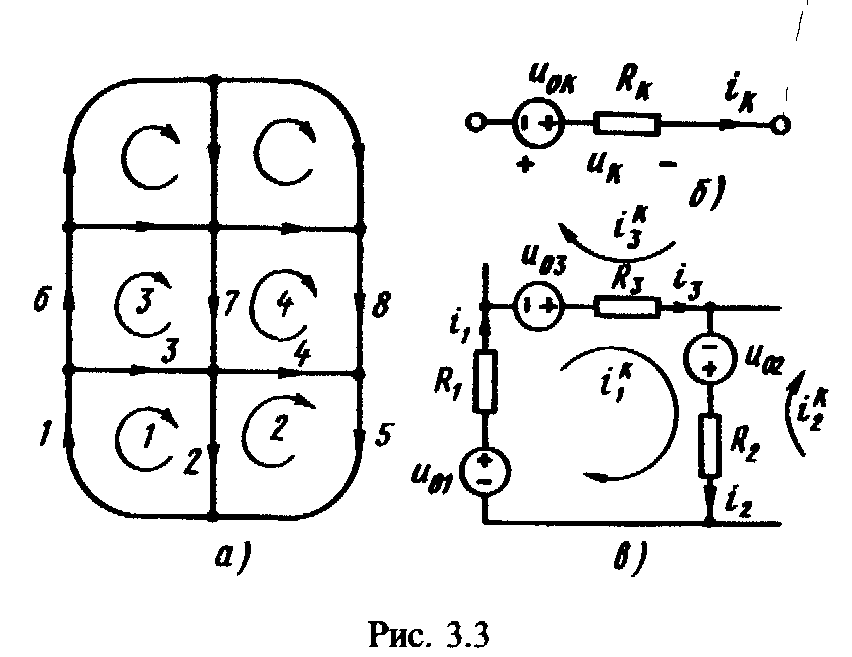
Строка этой матрицы показывает, между какими узлами присоединена каждая ветвь.

Если задана матрица соединений, то всегда можно построить соответствующий граф. Для этого, расположив точки, обозначающие узлы, следует соединить их попарно ветвями. Номера и направление ветвей определяются ненулевыми эле­ментами столбцов матрицы соединения.

## 2. Уравнения контурных токов

Метод контурных токов применим к цепям с планарным графом (рис. 3.3, а). В качестве переменных принимают замкну­тые контурные токи, проходящие по ветвям, образующим все внутренние ячейки графа.

Если намечать контуры – периметры ячеек – по порядку, начиная с одного края цепи, то легко убедиться, что в каждый последующий контур вносится новая ветвь, не вошедшая в предыдущие контуры. Отсюда следует, что уравнения равно­весия напряжений в таких контурах будут линейно независимы. Число внутренних ячеек равно nx=nв-ny+1.



Каждой ячейке при­писывается один кон­турный ток, замыкаю­щийся по ветвям, об­разующим ячейку. Об­щее число перемен­ных – контурных токов – равно числу ячеек. Направления всех контурных токов принимают одинаковыми – по часовой стрел­ке. Как видно из рис. 3.3, a, по каждой ветви цепи, за исключением периферийных ветвей, замыкаются два контурных тока, направленные в противоположные стороны. Запишем уравнения соединений.

1. Приравнивая нулю суммы напряжений ветвей всех неза­висимых контуров (ячеек), имеем n*х* уравнений по **ЗНК** .
2. Выражая ток каждой ветви через разность двух (в общем случае) замыкающихся по смежным контурам контурных токов, получим nв уравнений по **ЗТК**:

.

Как видно, токи всех ветвей, т. е. поведение всей цепи, полностью определяются n*х* контурными токами, число кото­рых меньше числа ветвей.

Запишем уравнения ветвей. Положим для удобства, что выполнено преобразование всех источников тока и цепь содержит только источники напряжения. Примем для общности каждую ветвь состоящей из последовательного соединения резистивного элемента и источника напряжения (рис. *3.3,6).* Уравнение такой составной ветви имеет вид:

.

Для получения уравнений относительно выбранных перемен­ных необходимо:

1. с помощью уравнений ветвей  в уравнениях равновесия напряжений заменить напряжения всех ветвей токами;
2. токи ветвей в получившейся системе заменить, согласно , контурными токами.

Получим уравнения для одной из ячеек, например первой (рис. 3.3, в), образованной тремя ветвями. Основным уравнени­ем равновесия напряжений в первом контуре будет:

u1+u2+u3=0 (\*)

Токи ветвей ячейки:

.(\*\*)

Уравнения ветвей:

 (\*\*\*)

Из трех систем уравнений (\*), (\*\*), (\*\*\*) необходимо получить уравнение, содержащее только искомые контурные токи. В соответствии со сказанным с помощью (\*\*\*) заменяем в основном уравнении (\*) напряжения на токи ветвей, которые затем выражаем через контурные токи согласно (\*\*):

,

После группировки имеем:



Первое слагаемое здесь представляет сумму напряжений всех резистивных ветвей контура только от собственного контурного тока в отсутствие токов других контуров (при их разрыве), а остальные слагаемые—напряжения ветвей контура от токов других контуров в отсутствие собственного контурного тока. В правую часть перенесены напряжения всех источников, входящих в контур.

Аналогичные уравнения получим для остальных контуров. Если число контуров равно *п,* топредположив для общности число ветвей каждой ячейки также равным *п,* можно записать систему уравнений контурных токов:



Коэффициент *Rkk—*собственное сопротивление контура, рав­ное сумме сопротивлений всех ветвей ячейки, а коэффициент *Rik=Rki (i<>k) –* взаимное сопротивление контуров, равное сопротивлению общей для контуров i и k ветви, взятому с отрицательным знаком, которым учитываются встречные на­правления контурных токов в рассматриваемой ветви.

Каждое уравнение системы выражает условие равно­весия напряжений ветвей контура – резистивных и источников напряжения (в правой части). Слагаемое на главной диагонали  дает напряжение всех резистивных ветвей только от собственного контурного тока, а слагаемое Рkjij=ukj - напря­жение на взаимном сопротивлении контуров только от тока в j-м контуре.

Составление уравнений сводится к записи симметричной матрицы параметров контурных токов:



Вектора контурных напряжений источников, составляющие которых равны суммам напряжений источников в контурах:



При введении вектора искомых контурных токов уравнения (3.10) в матричной форме можно записать в виде:

.

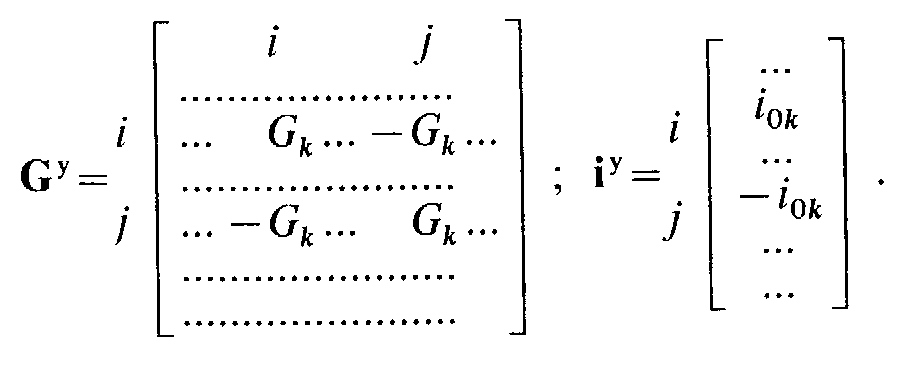
## 3. Алгоритм формирования узловых уравнений

Для ввода графа или соединений цепи производят последовательную нумерацию:

1. всех узлов от 1 до *п=пy-1* (опорному узлу присваивают нулевой номер);
2. всех ветвей от 1 до nв. Как указывалось, всю информацию о структуре графа содержит матрица соединений. Но вводить в память эту матрицу в виде двумерного массива нерационально из-за большого числа нулевых элементов. Поэтому структуру графа вводят с помощью таблицы соединений – одномерного массива троек целых чисел *(k, i, j),* где *k—*номер ветви; *i—*номер узла, откуда ветвь выходит; *j—*номер узла, куда ветвь входит. Тройки чисел дают ненулевые элементы aik=1 и аjk=-1 матрицы А. По заданной таблице соединений можно получить все необходимые для формирования уравнений матрицы.

Формировать узловые уравнения на ЭВМ можно перемножив произведение первых двух матриц па транспонированную матрицу соединений, получим матрицу узловых проводимостей; перемножение транспониро­ванной матрицы АТ на вектор токов источников дает вектор узловых токов. Действия над матрицами (транспонирование, сум­мирование, перемножение) легко программируется. Упомяну­тые матрицы являются разреженными, т. е. содержат много нуле­вых элементов. Поэтому приведенный алгоритм, включающий мно­го действий умножения на нулевой элемент, применять невыгодно.

Более рационально формировать матрицу **Gy** и вектор iy непосредственно - по мере поступления данных составных ветвей, исходя из смысла собственной и взаимной проводи­мостей. Вначале матрицу узловых проводимостей и вектор узловых токов принимают равными нулю: Gy=0 и iy=0*,* затем к ним добавляют элементы, вносимые каждой составной ветвью. Данные ветви (k, i, j) с проводимостью Gk и током источника тока iok войдут в собственные проводимости Gii, Gjj, узлов i, j как добавки Gk, во взаимные проводимости Gij, Gji как добавки Gk и в элементы вектора узловых токов iyi, iyi - как добавки ±i0k. Добавки ветви в матрицу проводимости Gy и вектор тока iy можно представить в виде:



Если ветвь присоединена к базисному узлу (j=0) то она внесет добавку *Gk,* только в собственную проводимость Gii и добавку i0k в составляющую iiy вектора узловых токов. Учет данных последней ветви завершает формирование узловых проводимостей и вектора узловых токов.

# Заключение

#### Таким образом в настоящей дипломной работе:

#### проведен литературный обзор по моделированию процессов в радиотехнических цепях, методов и средств расчетов токов и направлений в них;

#### построена математическая модель разветвленной цепи постоянного тока;

* на языке программирования Турбо Паскаль версии 7.0. в удобном графическом интерфейсе разработана программа построения и анализа электрической схемы цепи постоянного тока;
* приведены методические пояснения при работе с разработанной программой, а также расчетов и анализа электрических цепей;
* программа допускает без принципиальных изменений расширение её функциональных возможностей для расчетов цепей переменного тока, а также моделирования и анализа электромагнитных полей;
* результаты настоящей дипломной работы могут быть использованы в учебном процессе, при разработке электронной аппаратуры в научных лабораториях и на производстве.

В заключение хочу выразить благодарность своим научным руководителям: Цыпишка Дмитрию Ивановичу, Брагарь Луке Федеровичу и заведующиему кафедрой кафедрой общей физики и методики преподавания физики Стамову Ивану Григорьевичу, за методическую помощь, оказанную при написании дипломной работы.

# Использованная литература

1. К.С. Демирчан, П.А. Бутырин. Моделирование и машинный расчет электрических цепей. М., «Высшая школа», 1988г.
2. В. Нерретер. Расчет электрических цепей на ПЭВМ. М., «Энергоатомиздат», 1991г.
3. Пантюшин В.С. Сборник задач по электротехнике и основам электронники. М., «Высшая школа», 1979г.
4. П.Н. Махтанов. Основы анализа электрических цепей. Линейные цепи. М., «Высшая школа», 1990г.
5. «Электротехника». Под редакцией проф. В.С. Пантюшина. М., «Высшая школа», 1976г.
6. В.Г. Абрамов, Н.П. Трифонов, Г.Н. Трифонова. «Введение в язык Паскаль». М., «Наука», 1988г.
7. Ж.Джонс, К. Харроу. «Решение задач в системе Турбо Паскаль». М., «Финансы и статистика», 1991г.
8. К. Боон. «Паскаль для всех». М., «Энергоиздат», 1988г.
9. Д. Прайс. «Программирование на языке Паскаль». Практическое руководство. М., «Мир», 1987г.

# Приложение

I. Рисунки с видами экрана при работе с программой 20

Рисунок №1. Общий вид экрана. 20

Рисунок №2. Меню – Файл. 21

Рисунок №3. Открытие файла, сохраненного на диске. 22

Рисунок №4. Вид экрана с изображением схемы. 23

Рисунок №5. Вывод результата вычисления токов в ветвях схемы. 24

Рисунок №6. Просмотр направления токов в ветвях схемы. 25

Рисунок №7. Вид экрана при сохранении схемы в файл. 26

Рисунок №8. Меню – Окно. 27

Рисунок №9. Окно помощи выводимое на экран при нажатии клавиши F1. 29

II. Листинг программы на языке Паскаль. 30

1. Основная программа 30

2. Модуль с библиотекой элементов 36

3. Модуль вычисления токов ветвей 48

# I. Рисунки с видами экрана при работе с программой

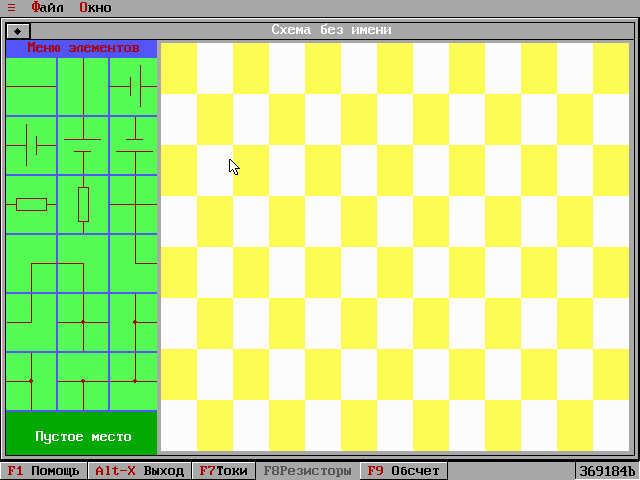
## Рисунок №1. Общий вид экрана.

Меню

Меню элементов

Рабочее поле

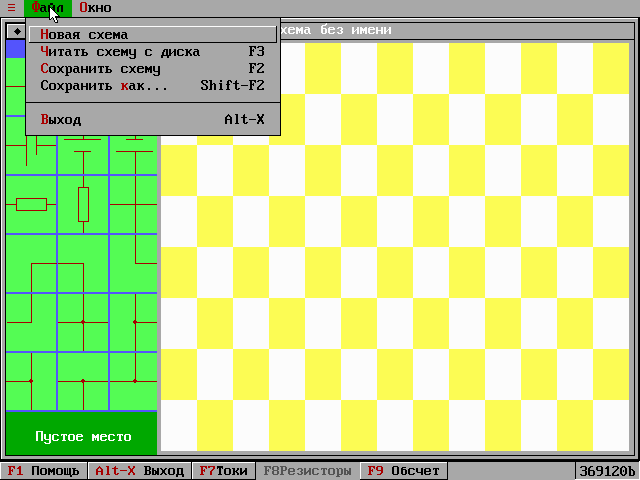
Указатель мыши



## Строка статуса

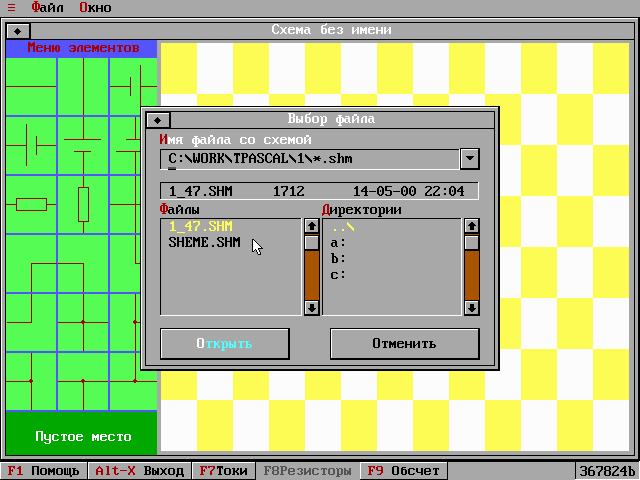
## Индекатор свободной памятиРисунок №2. Меню – Файл.

Меню – Файл



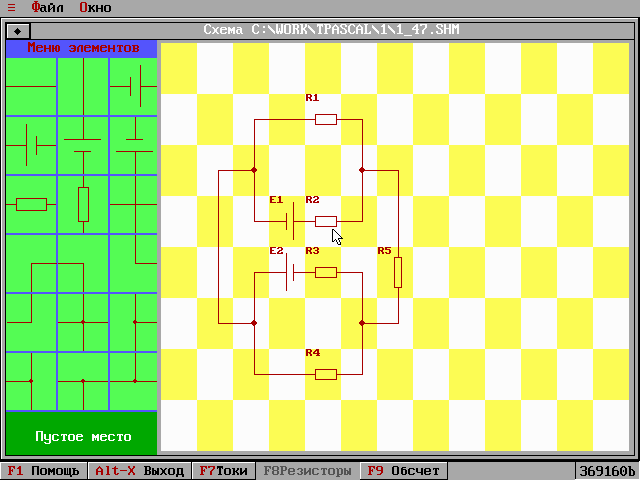
## Рисунок №3. Открытие файла, сохраненного на диске.

Открытие файла со схемой



## Рисунок №4. Вид экрана с изображением схемы.

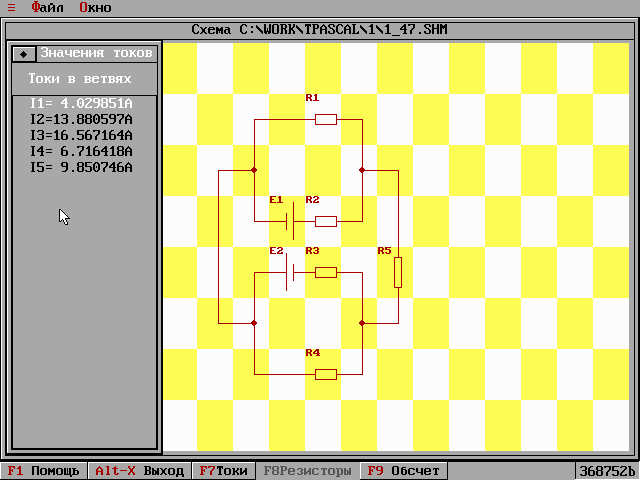
Изображение схемы



## Расчет схемы

## Рисунок №5. Вывод результата вычисления токов в ветвях схемы.

Результаты вычислений

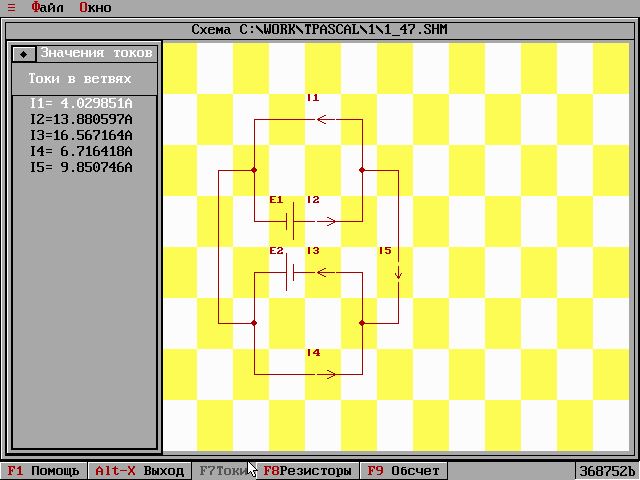


## Токи

## РезисторыРисунок №6. Просмотр направления токов в ветвях схемы.

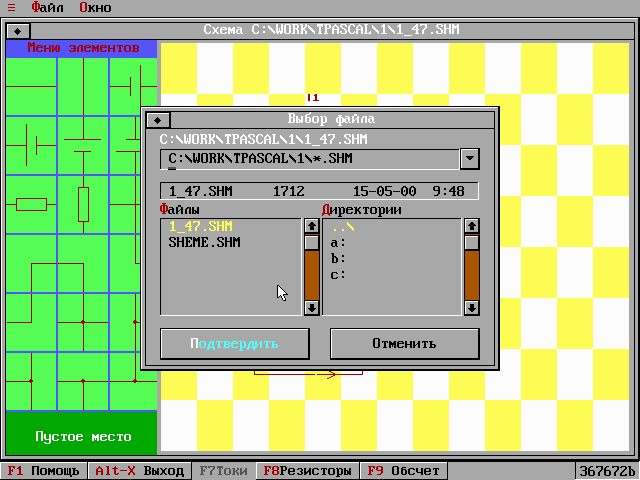
Значения токов

Направления токов



## Рисунок №7. Вид экрана при сохранении схемы в файл.

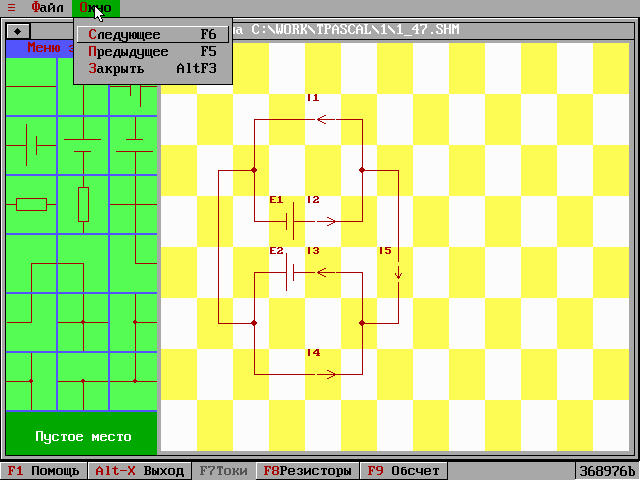
Сохранение схемы



## Рисунок №8. Меню – Окно.

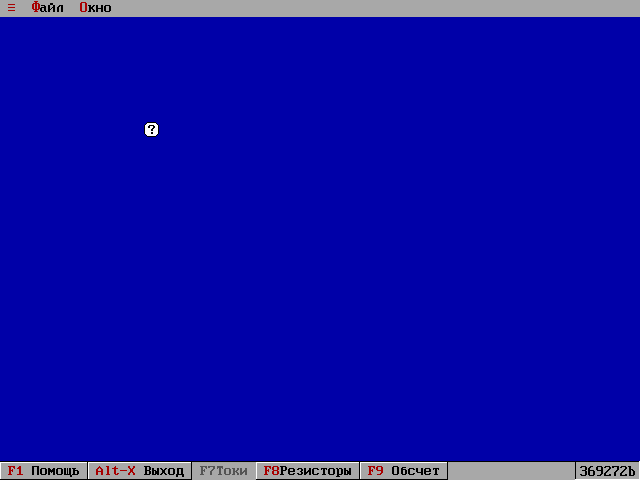
Работа с окнами

## Рисунок №9. Вид экрана при закрытии всех окон.



Указатель мыши

Меню

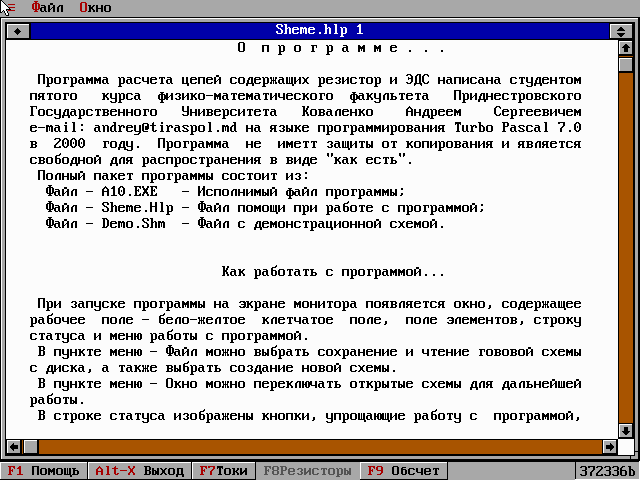


## Строка статуса

## Информация о свободной памятиРисунок №10. Окно помощи выводимое на экран при нажатии клавиши F1.

Закрытие окна

Окно с помощью программы



Горизонтальный скролинг

Вертикальный скролинг

# II. Листинг программы на языке Паскаль.

## 1. Основная программа

**Program** ***UzPotenc***; {Метод узловых потенциалов}{$F+,O+,X+,V-,R-,I-,S-}Uses

Crt, Applic1, Objects, Drivers, Dialogs, Views, Menus, App, StdDlg,

Fonts, HelpFile, MsgBox, TxtRead, WInDows, PalObj, Grv16, DemoHlp,

Types2;

Type TNewApp=Object(TMyApp)

Procedure ReCounte; Virtual;

End;

Var MyApp:TNewApp;

Type

PMyCollection=^TMyCollection;

TMyCollection=Object(TCollection)

Procedure FreeItem(Item:poInter); Virtual;

End;

**Procedure** ***CurView***; {Просмотр значений токов}

Var R,R1:TRect;

D:PDialog;

L:PListBox;

C:PMyCollection;

Sb:PScrollBar;

i:Integer;

s:String;

Begin

Sb:=Nil;

MyApp.ShemeWInDow^.GeTextentWIn(R);

R.B.X:=R.A.X+(R.B.X-R.A.X) Div 4;

D:=New(PDialog,Init(R,'Значения токов'));

D^.GeTextentWIn(R);

Inc(R.A.Y,CurrentFont^.Height\*2);

R1.Copy(R);

R1.A.X:=R1.B.X-CurrentFont^.Width\*2;

If RCount>(R.B.Y-R.A.Y) Div CurrentFont^.Height

Then

Begin

Sb:=D^.StAndardScrollBar(sbVertical+sbHAndleKeyBoard);

End;

C:=New(PMyCollection,Init(RCount,1));

For i:=1 To RCount Do

Begin

Str(abs(Currents[i]):9:6,s);

If i Div 10>0

Then C^.Insert(NewStr('I'+IntToStr(i)+'='+s+'A'))

Else C^.Insert(NewStr(' I'+IntToStr(i)+'='+s+'A'))

End;

L:=New(PListBox,Init(R,1,Sb));

L^.NewList(C);

D^.Insert(L);

R.B.Y:=R.A.Y;

Dec(R.A.Y,CurrentFont^.Height\*2);

D^.Insert(New(PLabel,Init(R,' Токи в ветвях',L)));

DeskTop^.Insert(D);

End;

**Procedure *TNewApp.ReCounte;***{Обсчет}

Var

i,j,k,l,m,Ii,Sizex,Index:Integer;

A:TElAr;

f1,f2:Boolean;

Ratio:Real;

**Function *Vetv1***(Ai,Aj,Ad:Integer):Boolean;

{Функция сохраняет в A ветвь от элемента (Ai,Aj) в направлении Ad (0-Up,1-Down,2-Left,3-Right и возвращает TRUE, если она содержит элементы}

Var i,j,k,l:Integer;

Flag1,Flag2:Boolean;

Begin

Flag1:=True;

Flag2:=False;

With A[Index] Do

Begin

Str:=Ai; Col:=Aj;

Num:=Sheme[Ai,Aj,2];

Typ:=Sheme[Ai,Aj,1];

End;

Inc(Index);

Case Ad Of

0: Begin i:=Ai+1; j:=Aj-1; End;

1: Begin i:=Ai-1; j:=Aj+1; End;

2: Begin i:=Ai-1; j:=Aj+1; End;

3: Begin i:=Ai+1; j:=Aj-1; End;

End;

While Flag1 And (i>0) And (j>0) And (i<=nS) And (j<=mS) And Not

(Sheme[i,j,1] In [0,14..18]) Do

Begin

If Sheme[i,j,1] In [3..8]

Then

Begin

Flag2:=True;

With A[Index] Do

Begin

Str:=i;

Col:=j;

Num:=Sheme[i,j,2];

Typ:=Sheme[i,j,1];

Case Ad Of

0:Dir:=Typ In [5,8];

1:Dir:=Typ=6;

2:Dir:=Typ=4;

3:Dir:=Typ In [3,7];

End;

End;

Inc(Index);

End;

Case Ad Of

0: Case Sheme[i,j,1] Of

2,5,6,8,9 : Dec(i);

10 : Begin Inc(j); Ad:=3; End;

11 : Begin Dec(j); Ad:=2; End;

Else Flag1:=False;

End;

1: Case Sheme[i,j,1] Of

2,5,6,8,9 : Inc(i);

12 : Begin Inc(j); Ad:=3; End;

13 : Begin Dec(j); Ad:=2; End;

Else Flag1:=False;

End;

2: Case Sheme[i,j,1] Of

1,3,4,7,9 : Dec(j);

10 : Begin Inc(i); Ad:=1; End;

12 : Begin Dec(i); Ad:=0; End;

Else Flag1:=False;

End;

3: Case Sheme[i,j,1] Of

1,3,4,7,9 : Inc(j);

13 : Begin Dec(i); Ad:=0; End;

11 : Begin Inc(i); Ad:=1; End;

Else Flag1:=False;

End;

End;

End;

If Sheme[i,j,1] In [14..18]

Then

Begin

With A[Index] Do

Begin

Str:=i;

Col:=j;

Num:=Sheme[i,j,2];

Typ:=Sheme[i,j,1];

End;

Inc(Index);

With A[Index] Do

Begin

Str:=0;

Col:=0;

Num:=0;

Typ:=0;

End;

Inc(Index);

End;

If Not Flag2

Then

Begin

For k:=1 To NoDecount Do

If (Nodes[k,1]=i) And (Nodes[k,2]=j)

Then l:=k;

NNum[l]:=NNum[Ii]; {Исключение накоротко замкнутых ветвей}

End;

Vetv1:=Flag2;

End;

**Function *ElEqu***(Var Src,Dst:TEl):Boolean; {Returns TRUE, If Src=Dst}

Begin

With Src Do

ElEqu:=(Str=Dst.Str)And(Col=Dst.Col)And(Typ=Dst.Typ)And(Num=Dst.Num);

End;

**Function *IsDiv***(Var Src:TEl):Boolean; {Returns TRUE, If Src - Divider}

Begin

With Src Do

IsDiv:=(Str=0)And(Col=0)And(Typ=0)And(Num=0);

End;

**Function *NextDiv***(i:Integer):Integer; {Поиск след. разд. элемента в массиве}

Begin

Repeat

Inc(i);

Until (i>Sizex) Or IsDiv(A[i]);

If i<=Sizex

Then NextDiv:=i

End;

**Function *PrevDiv***(i:Integer):Integer; {Поиск пред. разд. элемента в массиве}

Begin

Repeat

Dec(i);

Until (i<1) Or IsDiv(A[i]);

If i>=1

Then PrevDiv:=i

Else PrevDiv:=0;

End;

Begin

For i:=1 To nS\*mS Div 2 Do

For j:=1 To nS\*mS Div 2 Do

Equals[i,j]:=0;

For Ii:=1 To NoDecount Do

NNum[Ii]:=Ii;

Index:=1;

For Ii:=1 To NoDecount Do

Begin

Case Sheme[Nodes[Ii,1],Nodes[Ii,2],1] Of

14:Begin

Vetv1(Nodes[Ii,1],Nodes[Ii,2],0); Vetv1(Nodes[Ii,1],Nodes[Ii,2],1);

End;

15:Begin

Vetv1(Nodes[Ii,1],Nodes[Ii,2],0); Vetv1(Nodes[Ii,1],Nodes[Ii,2],1);

End;

16:Begin

Vetv1(Nodes[Ii,1],Nodes[Ii,2],0); Vetv1(Nodes[Ii,1],Nodes[Ii,2],1);

End;

17:Begin

Vetv1(Nodes[Ii,1],Nodes[Ii,2],1); Vetv1(Nodes[Ii,1],Nodes[Ii,2],2);

End;

18:Begin

Vetv1(Nodes[Ii,1],Nodes[Ii,2],0); Vetv1(Nodes[Ii,1],Nodes[Ii,2],2);

End;

End;

End;

Sizex:=Index-1;

{Оставляет нужные ветви}

i:=1;

While i<=Sizex Do

Begin

j:=0;

f1:=True;

While (i+j<=Sizex) And f1 Do

Begin

k:=NextDiv(i+j);

If ElEqu(A[k-1],A[i])And ElEqu(A[k-2],A[i+1])

Then

Begin

f1:=False;

l:=PrevDiv(k);

For m:=0 To Sizex-k Do

A[l+m]:=A[k+m];

Sizex:=Sizex-(k-l);

i:=NextDiv(i)+1;

If i=1

Then i:=Sizex+1;

End

Else

j:=k-i;

End;

End;

i:=0;

{Исключает пустые ветви}

While i<=Sizex Do

Begin

j:=NextDiv(i);

If j-i=3

Then

Begin

For k:=1 To Sizex-j Do

End;

If j<>0

Then i:=j

Else i:=Sizex+1;

End;

{Считаем сколько узлов с учётом соединений}

NCount:=NoDecount;

For i:=1 To NoDecount Do

If NNum[i]<>i

Then Dec(NCount);

If NCount<>NoDecount

Then

For i:=1 To NoDecount Do

Begin

j:=0;

For k:=1 To NoDecount Do

If NNum[k]=i

Then j:=1;

If j=0

Then

For k:=1 To NoDecount Do

If NNum[k]>i

Then Dec(NNum[k]);

End;

i:=1;

j:=0;

Repeat

Inc(j);

k:=NextDiv(i);

With Brunches[j] Do

Begin

AEDS:=0;

ARes:=0;

For l:=i To k Do

With A[l] Do

Case Typ Of

3..6: If Dir

Then EDS:=AEDS+EDS[Str,Col]

Else EDS:=AEDS-EDS[Str,Col];

7..8: ARes:=ARes+abs(Res[Str,Col]);

End;

FromN:=NNum[A[i].Num];

If k<>0

Then

Begin

ToN:=NNum[A[k-1].Num];

i:=k+1;

End

Else

Begin

ToN:=NNum[A[Sizex-1].Num];

i:=Sizex+1;

End;

End;

Until i>Sizex;

BrunchCount:=j;

{Заполняем систему}

For i:=1 To BrunchCount Do

With Brunches[i] Do

Begin

Equals[FromN,FromN]:=Equals[FromN,FromN]+1/ARes;

Equals[ToN,NCount+1]:=Equals[ToN,NCount+1]+AEDS/ARes;

End;

{Решаем систему}

For i:=2 To NCount Do

Begin

Ratio:=Equals[i,i];

For j:=2 To NCount+1 Do

Equals[i,j]:=Equals[i,j]/Ratio;

For k:=2 To NCount Do

If k<>i

For i:=1 To NCount+1 Do

Begin

Equals[1,i]:=0;

Equals[i,1]:=0;

End;

{После решения расставляем токи}

For i:=1 To RCount Do

Begin

j:=1;

While (j<=Sizex) And Not ((A[j].Typ In [7,8]) And (A[j].Num=i)) Do

Inc(j); k:=0; l:=j;

Repeat

k:=k+1; j:=PrevDiv(j);

Until j=0;

With Brunches[k] Do

Begin

Currents[i]:=(AEDS-Equals[ToN,NCount+1]+Equals[FromN,NCount+1])/ARes;

If Not A[l].Dir

Then Currents[i]:=-Currents[i];

End;

End;

CurView;

End;

**Procedure *TMyCollection.FreeItem***;

Begin

If Item<>Nil

Then DisposeStr(PString(Item));

End;

**BEGIN**

MyApp.Init;

MyApp.Run;

MyApp.Done;

**END**.

## 2. Модуль с библиотекой элементов

**Unit *Types2***;

Interface

Uses

Crt,

Objects, Drivers, Dialogs, Views, Menus, App, StdDlg,

Fonts, HelpFile, MsgBox, TxtRead, WInDows,

PalObj, Grv16, DemoHlp;

Const

nS=8;

mS=13;

Sx:Integer = 50;

Sy:Integer = 40;

Sx1:Integer=20;

Sy1:Integer=20;

cmMemoViewChange = 1001;

CurrentElement:Byte=0;

IsResist:Boolean=True; {If True - resistOrs, Else - currents}

Type

TSheme=Array [1..nS,1..mS,1..2] Of Byte; {Массив сдержит схему}

TNodes=Array [1..nS\*mS,1..2] Of Byte; {Массив содержит координаты всех

узловых элементов (i,j)}

TElems=Array [1..nS,1..mS] Of Real; {Содержит элементы значения}

TCurrents=Array [1..nS\*mS] Of Real; {Токи}

TNNum=Array [1..nS\*mS] Of Byte; {Номера узлов}

PEl=^TEl; {Элемент}

TEl=recOrd

Str,Col:Byte;{строка, столбец}

Typ:Byte;{тип}

Num:Byte;{номер}

Dir:Boolean;

End;

TBrunch=recOrd {Ветвь}

FromN,ToN:Byte;

ARes,AEDS:Real;

End;

TElAr=Array [1..2\*mS\*nS] Of TEl; {Элементы}

TBrunches=Array[1..mS\*nS] Of TBrunch; {Ветви}

TEquals=Array[1..mS\*nS Div 2,1..mS\*nS Div 2] Of Real; {Уравнения}

PToolBar = ^TToolBar;

TToolBar = Object(TView)

ConstructOr Init(Var R: TRect);

**Procedure *Draw***; Virtual;

**Procedure *HAndleEvent***(Var Event:TEvent); Virtual; {Реагирование на события}

End;

PMemoView = ^TMemoView;

TMemoView = Object(TView)

ConstructOr Init(Var Bounds: TRect);

**Procedure *HAndleEvent***(Var Event: TEvent); Virtual;

**Procedure *Draw***; Virtual;

End;

{П- указатель, Т - тип}

PShemeView = ^TShemeView;

TShemeView = Object(TView)

ConstructOr Init(Var R: TRect);

**Procedure *Draw***; Virtual;

**Procedure *HAndleEvent***(Var Event:TEvent); Virtual;

End;

PShemeWIn = ^TShemeWIn;

TShemeWIn = Object(TDialog)

ConstructOr Init(Var R:TRect);

**Function *ElMatter***(IsEDS:Boolean):Real; {Окно ввода значений}

DestructOr Done; Virtual;

End;

Var

Sheme:TSheme;

Nodes:TNodes;

EDS,Res:TElems;

Currents:TCurrents; {Токи}

NCount,NoDecount,ECount,RCount:Integer;

{Реално узлов, Узловых эл-тов, Колво ЭДС и Кол-во Рез.}

Changed:Boolean;

Exist:Boolean;

SetPhase:Boolean;

NNum:TNNum;

Brunches:TBrunches;

{Ветви}

BrunchCount:Integer;

{Кол-во}

Equals:TEquals;

**Function *IntToStr***(i:longInt):String;

**Procedure *ElNumbers***(Var ASheme:TSheme);

**Procedure *InitSheme***(Var ASheme:TSheme);

*Implementation*

**Procedure *InitSheme***(Var ASheme:TSheme);

{Зануляет текущую схему. Вызывается при старте и команде ОЧИСТИТЬ}

Var i,j,k:Integer;

Begin

For i:=1 To nS Do

For j:=1 To mS Do

For k:=1 To 2 Do

Begin

ASheme[i,j,k]:=0;

EDS[i,j]:=0;

Res[i,j]:=0;

End;

End;

ConstructOr TMemoView.Init(Var Bounds: TRect);

Begin

TView.Init(Bounds);

EventMask:= EventMask Or evBroadCast;

Options := OfPreProcess;

End;

**Procedure *TMemoView.HAndleEvent***(Var Event: TEvent);

Begin

Inherited HAndleEvent(Event);

With Event Do

If (What =evBroadCast)And(CommAnd=cmMemoViewChange)

Then DrawView

Else Exit;

ClearEvent(Event);

End;

**Procedure *TMemoView.Draw***;

Var

R: TRect;

S: String;

Begin

SetColOr(7);

FillRect(1, 1, Pred(Size.X), Pred(Size.Y));

GeTextent(R);

With R Do DrawFrame(A, B, OfWhiteRight);

Str(MemAvail:6, S);

SetColOr(0);

WriteStr(5, 3, S + 'b');

End;

ConstructOr TToolBar.Init(Var R: TRect);

Begin

Inherited Init(R);

GrowMode:= GrowMode Or (gfGrowHiX+gfGrowHiY);

End;

**Procedure *TToolBar.Draw***;

Var

i,j: Integer;

**Procedure *ElDraw***(Ax,Ay:Integer; An:Byte);

**Procedure *\_1***(x,y:Integer);

Begin

plotlIne (x,y+Sy Div 2,x+Sx,y+Sy Div 2);

End;

**Procedure *\_2***(x,y:Integer);

Begin

PlotLIne (x+sx Div 2,y,x+sx Div 2,y+sy);

End;

**Procedure *\_9***(x,y:Integer);

Begin

PlotLIne (x,y+sy Div 2,x+sx,y+sy Div 2);

PlotLIne (x+sx Div 2,y,x+sx Div 2,y+sy);

End;

*{ Procedure \_3(x,y:Integer);*

*Begin*

*PlotLIne (x,y+Sy Div 2,x+Sx Div 5,y+Sy Div 2);*

*PlotLIne (x+Sx\*4 Div 5,y+Sy Div 2,x+Sx,y+Sy Div 2);*

*ThickCircle(x+Sx Div 2,y+Sy Div 2,sx\*2 Div 6,1);*

*PlotLIne (x+Sx Div 4,y+Sy Div 2,x+Sx\*3 Div 4,y+Sy Div 2);*

*PlotLIne (x+Sx\*3 Div 4,y+Sy Div 2,x+Sx Div 2,y+Sy\*13 Div 20);*

*PlotLIne (x+Sx\*3 Div 4,y+Sy Div 2,x+Sx Div 2,y+Sy\*7 Div 20);*

*End;*

*Procedure \_4(x,y:Integer);*

*Begin*

*PlotLIne (x,y+sy Div 2,x+sx Div 5,y+sy Div 2);*

*PlotLIne (x+sx\*4 Div 5,y+sy Div 2,x+sx,y+sy Div 2);*

*ThickCircle(x+sx Div 2,y+sy Div 2,sx\*2 Div 6,1);*

*PlotLIne (x+sx Div 4,y+sy Div 2,x+sx\*3 Div 4,y+sy Div 2);*

*PlotLIne (x+sx Div 4,y+sy Div 2,x+sx Div 2,y+sy\*13 Div 20);*

*PlotLIne (x+sx Div 4,y+sy Div 2,x+sx Div 2,y+sy\*7 Div 20);*

*End;*

*Procedure \_5(x,y:Integer);*

*Begin*

*PlotLIne (x+sx Div 2,y,x+sx Div 2,y+sy\*2 Div 10);*

*PlotLIne (x+sx Div 2,y+sy\*8 Div 10,x+sx Div 2,y+sy);*

*ThickCircle(x+sx Div 2,y+sy Div 2,sx\*2 Div 6,1);*

*PlotLIne (x+sx Div 2,y+sy Div 4,x+sx Div 2,y+sy\*3 Div 4);*

*PlotLIne (x+sx Div 2,y+sy Div 4,x+sx\*13 Div 20,y+sy Div 2);*

*PlotLIne (x+sx Div 2,y+sy Div 4,x+sx\*7 Div 20,y+sy Div 2);*

*End;*

*Procedure \_6(x,y:Integer);*

*Begin*

*PlotLIne (x+sx Div 2,y,x+sx Div 2,y+sy\*2 Div 10);*

*PlotLIne (x+sx Div 2,y+sy\*8 Div 10,x+sx Div 2,y+sy);*

*ThickCircle(x+sx Div 2,y+sy Div 2,sx\*2 Div 6,1);*

*PlotLIne (x+sx Div 2,y+sy Div 4,x+sx Div 2,y+sy\*3 Div 4);*

*PlotLIne (x+sx Div 2,y+sy\*3 Div 4,x+sx\*13 Div 20,y+sy Div 2);*

*PlotLIne (x+sx Div 2,y+sy\*3 Div 4,x+sx\*7 Div 20,y+sy Div 2);*

*End;}*

**Procedure *\_3***(x,y:Integer);

Begin

PlotLIne (x,y+Sy Div 2,x+Sx\*2 Div 5,y+Sy Div 2);

PlotLIne (x+Sx\*3 Div 5,y+Sy Div 2,x+Sx,y+Sy Div 2);

PlotLIne (x+Sx\*3 Div 5,y+Sy Div 8,x+Sx\*3 Div 5,y+Sy\*7 Div 8);

PlotLIne (x+Sx\*2 Div 5,y+Sy Div 3,x+Sx\*2 Div 5,y+Sy\*2 Div 3);

End;

**Procedure *\_4***(x,y:Integer);

Begin

PlotLIne (x,y+Sy Div 2,x+Sx\*2 Div 5,y+Sy Div 2);

PlotLIne (x+Sx\*3 Div 5,y+Sy Div 2,x+Sx,y+Sy Div 2);

PlotLIne (x+Sx\*2 Div 5,y+Sy Div 8,x+Sx\*2 Div 5,y+Sy\*7 Div 8);

PlotLIne (x+Sx\*3 Div 5,y+Sy Div 3,x+Sx\*3 Div 5,y+Sy\*2 Div 3);

End;

**Procedure *\_5***(x,y:Integer);

Begin

PlotLIne (x+Sx Div 2,y,x+Sx Div 2,y+Sy\*2 Div 5);

PlotLIne (x+Sx Div 2,y+Sy\*3 Div 5,x+Sx Div 2,y+Sy);

PlotLIne (x+Sx Div 8,y+Sy\*2 Div 5,x+Sx\*7 Div 8,y+Sy\*2 Div 5);

PlotLIne (x+Sx Div 3,y+Sy\*3 Div 5,x+Sx\*2 Div 3,y+Sy\*3 Div 5);

End;

**Procedure *\_6***(x,y:Integer);

Begin

PlotLIne (x+Sx Div 2,y,x+Sx Div 2,y+Sy\*2 Div 5);

PlotLIne (x+Sx Div 2,y+Sy\*3 Div 5,x+Sx Div 2,y+Sy);

PlotLIne (x+Sx Div 8,y+Sy\*3 Div 5,x+Sx\*7 Div 8,y+Sy\*3 Div 5);

PlotLIne (x+Sx Div 3,y+Sy\*2 Div 5,x+Sx\*2 Div 3,y+Sy\*2 Div 5);

End;

**Procedure *\_7***(x,y:Integer);

Begin

PlotLIne(x,y+Sy Div 2,x+sx Div 5,y+Sy Div 2);

PlotLIne(x+sx\*4 Div 5,y+Sy Div 2,x+sx,y+Sy Div 2);

PlotLIne(x+sx Div 5,y+Sy\*12 Div 20,x+sx\*4 Div 5,y+Sy\*12 Div 20);

PlotLIne(x+sx\*4 Div 5,y+Sy\*12 Div 20,x+sx\*4 Div 5,y+Sy\*8 Div 20);

PlotLIne(x+sx\*4 Div 5,y+Sy\*8 Div 20,x+sx Div 5,y+Sy\*8 Div 20);

PlotLIne(x+sx Div 5,y+Sy\*8 Div 20,x+sx Div 5,y+Sy\*12 Div 20);

End;

**Procedure *\_8***(x,y:Integer);

Begin

PlotLIne(x+Sx Div 2,y,x+Sx Div 2,y+Sy Div 5);

PlotLIne(x+Sx Div 2,y+Sy\*4 Div 5,x+Sx Div 2,y+Sy);

PlotLIne(x+Sx\*12 Div 20,y+Sy Div 5,x+Sx\*12 Div 20,y+Sy\*4 Div 5);

PlotLIne(x+Sx\*12 Div 20,y+Sy\*4 Div 5,x+Sx\*8 Div 20,y+Sy\*4 Div 5);

PlotLIne(x+Sx\*8 Div 20,y+Sy\*4 Div 5,x+Sx\*8 Div 20,y+Sy Div 5);

PlotLIne(x+Sx\*8 Div 20,y+Sy Div 5,x+Sx\*12 Div 20,y+Sy Div 5);

End;

**Procedure *\_0***(x,y:Integer);

Begin

End;

**Procedure *\_10***(x,y:Integer);

Begin

PlotLIne(x+sx,y+sy Div 2,x+sx Div 2,y+sy Div 2);

PlotLIne(x+sx Div 2,y+sy Div 2,x+sx Div 2,y+sy);

End;

**Procedure *\_11***(x,y:Integer);

Begin

PlotLIne(x,y+sy Div 2,x+sx Div 2,y+sy Div 2);

PlotLIne(x+sx Div 2,y+sy Div 2,x+sx Div 2,y+sy);

End;

**Procedure *\_12***(x,y:Integer);

Begin

PlotLIne(x+sx Div 2,y,x+sx Div 2,y+sy Div 2);

PlotLIne(x+sx Div 2,y+sy Div 2,x+sx,y+sy Div 2);

End;

**Procedure *\_13***(x,y:Integer);

Begin

PlotLIne(x+sx Div 2,y,x+sx Div 2,y+sy Div 2);

PlotLIne(x+sx Div 2,y+sy Div 2,x,y+sy Div 2);

End;

**Procedure *\_14***(x,y:Integer);

Begin

PlotLIne (x,y+sy Div 2,x+sx,y+sy Div 2);

PlotLIne (x+sx Div 2,y,x+sx Div 2,y+sy);

FillCircle(x+sx Div 2,y+sy Div 2,2);

End;

**Procedure *\_15***(x,y:Integer);

Begin

PlotLIne (x+sx Div 2,y+sy Div 2,x+sx,y+sy Div 2);

PlotLIne (x+sx Div 2,y,x+sx Div 2,y+sy);

FillCircle(x+sx Div 2,y+sy Div 2,2);

End;

**Procedure *\_16***(x,y:Integer);

Begin

PlotLIne (x,y+sy Div 2,x+sx Div 2,y+sy Div 2);

PlotLIne (x+sx Div 2,y,x+sx Div 2,y+sy);

FillCircle(x+sx Div 2,y+sy Div 2,2);

End;

**Procedure *\_17***(x,y:Integer);

Begin

PlotLIne (x,y+sy Div 2,x+sx,y+sy Div 2);

PlotLIne (x+sx Div 2,y+sy Div 2,x+sx Div 2,y+sy);

FillCircle(x+sx Div 2,y+sy Div 2,2);

End;

**Procedure *\_18***(x,y:Integer);

Begin

PlotLIne (x,y+sy Div 2,x+sx,y+sy Div 2);

PlotLIne (x+sx Div 2,y,x+sx Div 2,y+sy Div 2);

FillCircle(x+sx Div 2,y+sy Div 2,2);

End;

Begin

If An=CurrentElement

Then

SetColOr(2)

Else

SetColOr(10);

FillRect(Ax,Ay,Sx,Sy);

SetColOr(4);

Case An Of

1:\_1(Ax,Ay); 2:\_2(Ax,Ay); 3:\_3(Ax,Ay); 4:\_4(Ax,Ay); 5:\_5(Ax,Ay); 6:\_6(Ax,Ay); 7:\_7(Ax,Ay); 8:\_8(Ax,Ay);

9:\_9(Ax,Ay); 10:\_10(Ax,Ay); 11:\_11(Ax,Ay); 12:\_12(Ax,Ay);

13:\_13(Ax,Ay); 14:\_14(Ax,Ay); 15:\_15(Ax,Ay); 16:\_16(Ax,Ay);

17:\_17(Ax,Ay); 18:\_18(Ax,Ay);

Else \_0(Ax,Ay);

End;

End;

Begin

With Size Do

Begin

Sx:=x Div 3 - 2; Sy:=y Div 7 - 2;

End;

SetColOr(9);

FillRect(0,0,Size.X,(Sy+2)\*6+CurrentFont^.Height+2);

SetColOr(4);

WriteStr((Size.X-14\*CurrentFont^.Width) Div 2, 0, 'Меню элементов');

For i:=1 To 6 Do

For j:=1 To 3 Do

ElDraw((j-1)\*(Sx+2),(i-1)\*(Sy+2)+CurrentFont^.Height+2,(i-1)\*3+j);

If CurrentElement=0

Then

SetColOr(2)

Else

SetColOr(10);

FillRect(0,(Sy+2)\*6+CurrentFont^.Height+2,Size.X,Size.Y);

SetColOr(15);

WriteStr((Size.X-12\*CurrentFont^.Width) Div 2,((Sy+2)\*6+

CurrentFont^.Height Div 2 +2 + Size.Y) Div 2, 'Пустое место');

End;

**Procedure *TToolBar.HAndleEvent***;

Var x,y:Integer;

Begin

Inherited HAndleEvent(Event);

If (Event.What=evMouseDown) And (Event.Buttons=mbLeftButton)

Then

Begin

x:=(Event.Where.X-CurrentFont^.Width-2) Div Sx;

y:=(Event.Where.Y-CurrentFont^.Height-2) Div Sy-1;

CurrentElement:=y\*3+x+1;

If Event.Where.Y>Sy\*7+CurrentFont^.Height+2

Then CurrentElement:=0;

DrawView;

ClearEvent(Event);

End;

End;

ConstructOr TShemeView.Init(Var R: TRect);

Begin

Inherited Init(R);

Font:=@Font8x8;

GrowMode:= GrowMode Or (gfGrowHiX+gfGrowHiY);

End;

**Procedure *TShemeView.Draw***;

Const

Special:Integer=2;

Var

i,j: Integer;

c:Byte;

**Procedure *ElDraw***(Ax,Ay:Integer; An,l:Byte);

**Procedure *\_1***(x,y:Integer);

Begin

plotlIne (x,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

End;

**Procedure *\_2***(x,y:Integer);

Begin

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1);

End;

**Procedure *\_9***(x,y:Integer);

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1);

End;

*{ Procedure \_3(x,y:Integer);*

*Begin*

*PlotLIne (x,y+Sy1 Div 2,x+Sx1 Div 5+Special,y+Sy1 Div 2);*

*PlotLIne (x+Sx1\*4 Div 5+Special,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);*

*ThickCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,Sx1\*2 Div 6,1);*

*PlotLIne (x+Sx1 Div 4+Special,y+Sy1 Div 2,x+Sx1\*3 Div 4+Special,y+Sy1 Div 2);*

*PlotLIne (x+Sx1\*3 Div 4+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1\*12 Div 20);*

*PlotLIne (x+Sx1\*3 Div 4+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1\*8 Div 20);*

*End;*

*Procedure \_4(x,y:Integer);*

*Begin*

*PlotLIne (x,y+Sy1 Div 2,x+Sx1 Div 5+Special,y+Sy1 Div 2);*

*PlotLIne (x+Sx1\*4 Div 5+Special,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);*

*ThickCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,Sx1\*2 Div 6,1);*

*PlotLIne (x+Sx1 Div 4+Special,y+Sy1 Div 2,x+Sx1\*3 Div 4+Special,y+Sy1 Div 2);*

*PlotLIne (x+Sx1 Div 4+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1\*12 Div 20);*

*PlotLIne (x+Sx1 Div 4+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1\*8 Div 20);*

*End;*

*Procedure \_5(x,y:Integer);*

*Begin*

*PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1\*3 Div 10);*

*PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*7 Div 10,x+Sx1 Div 2+Special,y+Sy1);*

*ThickCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,Sx1\*2 Div 6,1);*

*PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 8,x+Sx1 Div 2+Special,y+Sy1\*5 Div 8);*

*PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 8,x+Sx1\*12 Div 20+Special,y+Sy1 Div 2);*

*PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 8,x+Sx1\*8 Div 20+Special,y+Sy1 Div 2);*

*End;*

*Procedure \_6(x,y:Integer);*

*Begin*

*PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1\*3 Div 10);*

*PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*7 Div 10,x+Sx1 Div 2+Special,y+Sy1);*

*ThickCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,Sx1\*2 Div 6,1);*

*PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 8,x+Sx1 Div 2+Special,y+Sy1\*5 Div 8);*

*PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*5 Div 8,x+Sx1\*12 Div 20+Special,y+Sy1 Div 2);*

*PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*5 Div 8,x+Sx1\*8 Div 20+Special,y+Sy1 Div 2);*

*End;}*

**Procedure *\_3***(x,y:Integer);

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1\*2 Div 5+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1\*3 Div 5+Special,y+Sy1 Div 2,x+Sx1+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1\*3 Div 5+Special,y+Sy1 Div 8,x+Sx1\*3 Div 5+Special,y+Sy1\*7 Div 8);

PlotLIne (x+Sx1\*2 Div 5+Special,y+Sy1 Div 3,x+Sx1\*2 Div 5+Special,y+Sy1\*2 Div 3);

End;

**Procedure *\_4***(x,y:Integer);

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1\*2 Div 5+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1\*3 Div 5+Special,y+Sy1 Div 2,x+Sx1+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1\*2 Div 5+Special,y+Sy1 Div 8,x+Sx1\*2 Div 5+Special,y+Sy1\*7 Div 8);

PlotLIne (x+Sx1\*3 Div 5+Special,y+Sy1 Div 3,x+Sx1\*3 Div 5+Special,y+Sy1\*2 Div 3);

End;

**Procedure *\_5***(x,y:Integer);

Begin

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1\*2 Div 5);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 5,x+Sx1 Div 2+Special,y+Sy1);

PlotLIne (x+Sx1 Div 8+Special,y+Sy1\*2 Div 5,x+Sx1\*7 Div 8+Special,y+Sy1\*2 Div 5);

PlotLIne (x+Sx1 Div 3+Special,y+Sy1\*3 Div 5,x+Sx1\*2 Div 3+Special,y+Sy1\*3 Div 5);

End;

**Procedure *\_6***(x,y:Integer);

Begin

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1\*2 Div 5);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 5,x+Sx1 Div 2+Special,y+Sy1);

PlotLIne (x+Sx1 Div 8+Special,y+Sy1\*3 Div 5,x+Sx1\*7 Div 8+Special,y+Sy1\*3 Div 5);

PlotLIne (x+Sx1 Div 3+Special,y+Sy1\*2 Div 5,x+Sx1\*2 Div 3+Special,y+Sy1\*2 Div 5);

End;

**Procedure *\_7***(x,y:Integer);

Begin

If IsResist

Then

Begin

PlotLIne(x,y+Sy1 Div 2,x+Sx1 Div 5+Special,y+Sy1 Div 2);

PlotLIne(x+Sx1\*4 Div 5+Special,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

PlotLIne(x+Sx1 Div 5+Special,y+Sy1\*12 Div 20,x+Sx1\*4 Div 5+Special,y+Sy1\*12 Div 20);

PlotLIne(x+Sx1\*4 Div 5+Special,y+Sy1\*12 Div 20,x+Sx1\*4 Div 5+Special,y+Sy1\*8 Div 20);

PlotLIne(x+Sx1\*4 Div 5+Special,y+Sy1\*8 Div 20,x+Sx1 Div 5+Special,y+Sy1\*8 Div 20);

PlotLIne(x+Sx1 Div 5+Special,y+Sy1\*8 Div 20,x+Sx1 Div 5+Special,y+Sy1\*12 Div 20);

End

Else

If Currents[Sheme[i,j,2]]>0

Then

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1 Div 5+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1\*4 Div 5+Special,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 4+Special,y+Sy1 Div 2,x+Sx1\*3 Div 4+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1\*3 Div 4+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1\*12 Div 20);

PlotLIne (x+Sx1\*3 Div 4+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1\*8 Div 20);

End

Else

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1 Div 5+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1\*4 Div 5+Special,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 4+Special,y+Sy1 Div 2,x+Sx1\*3 Div 4+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 4+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1\*12 Div 20);

PlotLIne (x+Sx1 Div 4+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1\*8 Div 20);

End;

End;

**Procedure *\_8***(x,y:Integer);

Begin

If IsResist

Then

Begin

PlotLIne(x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1 Div 5);

PlotLIne(x+Sx1 Div 2+Special,y+Sy1\*4 Div 5,x+Sx1 Div 2+Special,y+Sy1);

PlotLIne(x+Sx1\*12 Div 20+Special,y+Sy1 Div 5,x+Sx1\*12 Div 20+Special,y+Sy1\*4 Div 5);

PlotLIne(x+Sx1\*12 Div 20+Special,y+Sy1\*4 Div 5,x+Sx1\*8 Div 20+Special,y+Sy1\*4 Div 5);

PlotLIne(x+Sx1\*8 Div 20+Special,y+Sy1\*4 Div 5,x+Sx1\*8 Div 20+Special,y+Sy1 Div 5);

PlotLIne(x+Sx1\*8 Div 20+Special,y+Sy1 Div 5,x+Sx1\*12 Div 20+Special,y+Sy1 Div 5);

End

Else

If Currents[Sheme[i,j,2]]>0

Then

Begin

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1\*3 Div 10);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*7 Div 10,x+Sx1 Div 2+Special,y+Sy1);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 8,x+Sx1 Div 2+Special,y+Sy1\*5 Div 8);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 8,x+Sx1\*12 Div 20+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 8,x+Sx1\*8 Div 20+Special,y+Sy1 Div 2);

End

Else

Begin

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1\*3 Div 10);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*7 Div 10,x+Sx1 Div 2+Special,y+Sy1);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*3 Div 8,x+Sx1 Div 2+Special,y+Sy1\*5 Div 8);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*5 Div 8,x+Sx1\*12 Div 20+Special,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1\*5 Div 8,x+Sx1\*8 Div 20+Special,y+Sy1 Div 2);

End;

End;

**Procedure *\_0***(x,y:Integer);

Begin

End;

**Procedure *\_10***(x,y:Integer);

Begin

PlotLIne(x+Sx1,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1 Div 2);

PlotLIne(x+Sx1 Div 2+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1);

End;

**Procedure *\_11***(x,y:Integer);

Begin

PlotLIne(x,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1 Div 2);

PlotLIne(x+Sx1 Div 2+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1);

End;

**Procedure *\_12***(x,y:Integer);

Begin

PlotLIne(x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1 Div 2);

PlotLIne(x+Sx1 Div 2+Special,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

End;

**Procedure *\_13***(x,y:Integer);

Begin

PlotLIne(x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1 Div 2);

PlotLIne(x,y+Sy1 Div 2,x+Sx1 Div 2+Special+1,y+Sy1 Div 2);

End;

**Procedure *\_14***(x,y:Integer);

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1);

FillCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,3);

End;

**Procedure *\_15***(x,y:Integer);

Begin

PlotLIne (x+Sx1 Div 2+Special,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1);

FillCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,3);

End;

**Procedure *\_16***(x,y:Integer);

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1 Div 2,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1);

FillCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,3);

End;

**Procedure *\_17***(x,y:Integer);

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 2+Special,y+Sy1 Div 2,x+Sx1 Div 2+Special,y+Sy1);

FillCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,3);

End;

**Procedure *\_18***(x,y:Integer);

Begin

PlotLIne (x,y+Sy1 Div 2,x+Sx1,y+Sy1 Div 2);

PlotLIne (x+Sx1 Div 2+Special,y,x+Sx1 Div 2+Special,y+Sy1 Div 2);

FillCircle(x+Sx1 Div 2+Special,y+Sy1 Div 2,3);

End;

Begin

Special:=Sx1 Div 10;

SetColOr(l);

FillRect(Ax,Ay,Sx1,Sy1);

SetColOr(4);

Case An Of

1:\_1(Ax,Ay); 2:\_2(Ax,Ay); 3:\_3(Ax,Ay); 4:\_4(Ax,Ay);

5:\_5(Ax,Ay); 6:\_6(Ax,Ay); 7:\_7(Ax,Ay); 8:\_8(Ax,Ay);

9:\_9(Ax,Ay); 10:\_10(Ax,Ay); 11:\_11(Ax,Ay); 12:\_12(Ax,Ay);

13:\_13(Ax,Ay); 14:\_14(Ax,Ay); 15:\_15(Ax,Ay); 16:\_16(Ax,Ay);

17:\_17(Ax,Ay); 18:\_18(Ax,Ay);

Else \_0(Ax,Ay);

End;

End;

Begin

C:= GetColOr(6);

{Определение цвета нормального текста}

SetColOr(C shr 4);

With Size Do

Begin

FillRect(0, 0, Size.X, Size.Y);

Sx1:=x Div mS;

Sy1:=y Div nS;

For i:=1 To nS Do

For j:=1 To mS Do

Begin

ElDraw((j-1)\*Sx1+(x-Sx1\*mS) Div 2 ,(i-1)\*Sy1+(y-Sy1\*nS) Div 2,Sheme[i,j,1],((i+j) mod 2)+14);

Case Sheme[i,j,1]Of

3,4,5,6:WriteStr((j-1)\*Sx1+(x-Sx1\*mS) Div 2 ,(i-1)\*Sy1+(y-Sy1\*nS) Div 2,'E'+IntToStr(Sheme[i,j,2]));

7,8: If IsResist

Then WriteStr((j-1)\*Sx1+(x-Sx1\*mS) Div 2 ,(i-1)\*Sy1+(y-Sy1\*nS) Div 2,'R'+IntToStr(Sheme[i,j,2]))

Else WriteStr((j-1)\*Sx1+(x-Sx1\*mS) Div 2 ,(i-1)\*Sy1+(y-Sy1\*nS) Div 2,'I'+IntToStr(Sheme[i,j,2]));

End; {Of Case}

End;

{ RestOreFont;}

End;

End;

**Procedure *TShemeView.HAndleEvent***;

Var x,y:Integer;

Begin

Inherited HAndleEvent(Event);

If (Event.What=evMouseDown) And (Event.Buttons=mbLeftButton)

Then

Begin

x:=(Event.Where.X-Sx1\*3 Div 8-(Size.X-Sx1\*mS) Div 2) Div Sx1-3;

y:=(Event.Where.Y-(Size.Y-Sy1\*nS) Div 2) Div Sy1;

Case Sheme[y,x,1] Of

3..6: EDS[y,x]:=0;

7..8: Res[y,x]:=0;

End;

Sheme[y,x,1]:=CurrentElement;

Changed:=True;

ElNumbers(Sheme);

DrawView;

Case CurrentElement Of

3..6: EDS[y,x]:=PShemeWIn(Owner)^.ElMatter(True);

7..8: Res[y,x]:=PShemeWIn(Owner)^.ElMatter(False);

End;

ClearEvent(Event);

End;

End;

**Function *IntToStr***(I: LongInt): String;

{ Convert any Integer Type To a String }

Var S: String[11];

Begin

Str(I, S);

IntToStr:= S;

End;

**Procedure *ElNumbers***(Var ASheme:TSheme);

{Нумерует элементы схемы (ЭДС, резисторы и узловые элементы для служебных

целей).Вызывается когда схема готова}

Var i,j:Integer;

nE,nR,nN:Byte;

Begin

nE:=0;nR:=0;nN:=0;

For j:=1 To mS Do

For i:=1 To nS Do

Case ASheme[i,j,1] Of

3,4,5,6: Begin {ЭДС} Inc(nE); ASheme[i,j,2]:=nE; End;

7,8: Begin {резистор} Inc(nR); ASheme[i,j,2]:=nR; End;

14..18: Begin Inc(nN); ASheme[i,j,2]:=nN; Nodes[nN,1]:=i; Nodes[nN,2]:=j; End;

End; {Of Case}

ECount:=nE; RCount:=nR; NoDecount:=nN;

End;

ConstructOr TShemeWIn.Init;

Begin

Inherited Init(R, 'Схема без имени');

SetPhase:=True;

Exist:=True;

Options:= Options Or OfCentered;

DragMode:=0;

Palette:= wpCyanWInDow;

GeTextentWIn(R);

R.B.X:=R.A.X+(R.B.X-R.A.X) Div 4;

Insert(New(PToolBar, Init(R)));

GeTextentWIn(R);

R.A.X:=(R.B.X-R.A.X) Div 4;

Insert(New(PShemeView, Init(R)));

HelpCtx:= hcGraphic;

End;

**Function *TShemeWIn.ElMatter***;

Var R:TRect;

M:Real;

c:wOrd;

i:Integer;

D:PDialog;

L:PInputLIne;

s:String;

Begin

M:=0;

GeTextentWIn(R);

R.B.X:=R.A.X+(R.B.X-R.A.X) Div 4;

Inc(R.A.Y,CurrentFont^.Height\*5);

Dec(R.B.Y,CurrentFont^.Height\*10);

If IsEDS

Then s:='Напряжение'

Else s:='Сопртивление';

D:=New(PDialog,Init(R,s));

Inc(R.A.Y,CurrentFont^.Height\*3);

Inc(R.A.X,CurrentFont^.Width\*5);

Dec(R.B.X,CurrentFont^.Width\*5);

R.B.Y:=R.A.Y+CurrentFont^.Height\*1;

L:=New(PInputLIne,Init(R,10));

If D<>Nil

Then

Begin

D^.GeTextentWIn(R);

Inc(R.A.Y,CurrentFont^.Height Div 2);

Inc(R.A.X,CurrentFont^.Width);

Dec(R.B.X,CurrentFont^.Width\*4);

R.B.Y:=R.A.Y+CurrentFont^.Height;

L:=New(PInputLIne,Init(R,10));

R.A.X:=R.B.X+CurrentFont^.Width;

R.B.X:=R.A.X+CurrentFont^.Width\*3;

If IsEDS

Then s:='В'

Else s:='Ом';

D^.Insert(New(PStaticText,Init(R,s)));

D^.GeTextentWIn(R);

R.Move(CurrentFont^.Width\*2,CurrentFont^.Height\*2);

R.B.Y:=R.A.Y+CurrentFont^.Height;

R.B.X:=R.A.X+CurrentFont^.Width\*15;

D^.Insert(New(PButton,Init(R,'O~k~',cmOk,bfDefault)));

If L<>Nil

Then

D^.Insert(L);

c:=DeskTop^.ExecView(D);

If c<>cmCancel

Then

Begin

If L<>Nil

Then

Begin L^.GetData(s); Dispose(L,Done); End;

i:=0;

val(s,M,i);

End;

If D<>Nil

Then

Dispose(D,Done);

End;

ElMatter:=M;

End;

DestructOr TShemeWIn.Done;

Begin

Inherited Done;

Exist:=False;

End;

**END**.

## 3. Модуль вычисления токов ветвей

**Unit *Applic1***;

{$F+,O+,X+,V-,R-,I-,S-}

Interface

Uses

Crt,

Objects, Drivers, Dialogs, Views, Menus, App, StdDlg,

Fonts, HelpFile, MsgBox, TxtRead, WInDows,

PalObj, Grv16, DemoHlp, Types2;

Const

cmAbout = 100;

cmReCounte = 101;

cmTxtWInDow = 102;

cmDialog = 103;

cmDemOfonts = 104;

cmDemoPic = 105;

cmWInWIn = 106;

cmCur = 107;

cmRes = 108;

cmIdle = 6000;

HelpName:String ='Sheme.hlp';

Var

ValDel: LongInt;

Ticks: WOrd absolute $40:$6C; { BIOS Timer ticks counter }

Type

TMyApp = Object(TApplication)

MemoAvail: LongInt; {Свободная мем}

ShemeWInDow: PShemeWIn; {Окно}

ShemeName: String; {Имя схемы}

ConstructOr Init; {Добавление нового }

Procedure HAndleEvent(Var Event: TEvent); Virtual;

Procedure InitMenuBar; Virtual;

Procedure InitStatusLIne; Virtual;

Procedure ReCounte; Virtual;

Procedure About;

Procedure HlpWInDow;

Procedure NewSheme;

Procedure OpenSheme;

Procedure SaveSheme;

Procedure SaveShemeAs;

Procedure Idle; Virtual; {Обновление показ. памяти}

End;

Implementation

ConstructOr TMyApp.Init;

Var

R: TRect;

Begin

Inherited Init;

InitSheme(Sheme);

ShemeName:='';

Changed:=False;

StatusLIne^.GetBounds(R);

R.A.X:= R.B.X - 65;

Insert(New(PMemoView, Init(R)));

MemoAvail:= MemAvail;

ValDel:= Ticks;

DeskTop^.GeTextent(R);

ShemeWInDow:=New(PShemeWIn,Init(R));

DeskTop^.Insert(ShemeWInDow);

DisableCommAnds([cmRes]);

EnableCommAnds([cmCur]);

End;

**Procedure *TMyApp.Idle***;

**Function *IsTileable***(P: PView): Boolean;

Begin

IsTileable:= (P^.Options And OfTileable) <> 0;

End;

Begin

Inherited Idle;

Message(@Self, evBroadCast, cmIdle, Nil);

If MemoAvail <> MemAvail Then Begin

Message(@Self, evBroadCast, cmMemoViewChange, Nil);

MemoAvail:= MemAvail;

End;

If Desktop^.FirstThat(@IsTileable) <> Nil

Then EnableCommAnds([cmTile, cmCascade])

Else DisableCommAnds([cmTile, cmCascade]);

End;

**Procedure *TMyApp.InitMenuBar***;

Var R: TRect;

Begin

GeTextent (R);

R.B.Y:= R.A.Y + CurrentFont^.Height + 1;

MenuBar:= New(PMenuBar, Init(R, NewMenu(

NewItem('~Ё~', '', kbAltSpace, cmAbout, hcMenu10,

NewSubMenu('~Ф~айл', hcMenu20, NewMenu(

NewItem('~Н~овая схема', '', kbNoKey, cmNew, hcNoConText,

NewItem('~Ч~итать схему с диска', 'F3', kbF3, cmOpen, hcNoConText,

NewItem('~C~охранить схему', 'F2', kbF2, cmSave, hcNoConText,

NewItem('Cохранить ~к~ак...', 'ShIft-F2', kbShIftF2, cmSaveAs, hcNoConText,

NewLIne(

NewItem('~В~ыход', 'Alt-X', kbAltX, cmQuit, hcNoConText,

Nil))))))),

NewSubMenu('~О~кно', hcMenu30, NewMenu(

NewItem('~С~ледующее', 'F6', kbF6, cmNext, hcNoConText,

NewItem('~П~редыдущее', 'F5', kbF5, cmPrev, hcNoConText,

NewItem('~З~акрыть', 'AltF3', kbAltF3, cmClose, hcNoConText,

Nil)))),

Nil)

)))));

MenuBar^.State:= MenuBar^.State Or sfActive;

End;

**Procedure *TMyApp.InitStatusLIne***;

Var R: TRect;

Begin

GeTextent(R);

R.A.Y:= R.B.Y - 19;

SetFont(@Font8x14);

StatusLIne:= New(PStatusLIne, Init(R,

NewStatusDef(0, $0FFF,

NewStatusKey('', kbAltF3, cmClose,

NewStatusKey('~F1~ Помощь', kbF1, cmHelp,

NewStatusKey('', kbF10, cmMenu,

NewStatusKey('~Alt-X~ Выход', kbAltX, cmQuit,

NewStatusKey('~F7~Токи', kbF7, cmCur,

NewStatusKey('~F8~Резисторы', kbF8, cmRes,

NewStatusKey('~F9~ Обсчет', kbF9, cmReCounte,

Nil))))))),

NewStatusDef($1000, $1001,

NewStatusKey('~Ctrl-'#24#25#26#27'~ Перемещение', kbNoKey, cmNo,

NewStatusKey('~Shft-'#24#25#26#27'~ Размер', kbNoKey, cmNo,

NewStatusKey('~'#17#217'~ Подтвердить', kbNoKey, cmNo,

NewStatusKey('~Esc~ Отменить', kbNoKey, cmNo,

Nil)))),

NewStatusDef($FFFE, $FFFF,

NewStatusKey('', kbAltF3, cmClose,

NewStatusKey('~Tab~ След. индекс', kbTab, cmNextTopic,

NewStatusKey('~ShIft-Tab~ Пред. индекс', kbShIftTab, cmPrevTopic,

NewStatusKey('~Esc~ Закрыть', kbEsc, cmClose,

Nil)))),

Nil)))

));

RestOreFont;

End;

**Procedure *TMyApp.HAndleEvent***(Var Event: TEvent);

Var

R: TRect;

P: PView;

Control: WOrd;

SavePalette: PaletteType;

Begin

Inherited HAndleEvent(Event);

If Event.What = evCommAnd Then

Begin

Case Event.CommAnd Of

cmNew : NewSheme;

cmOpen : OpenSheme;

cmSave : If ShemeName=''

Then SaveShemeAs

Else SaveSheme;

cmSaveAs : SaveShemeAs;

cmReCounte : ReCounte;

cmAbout : About;

cmHelp : HlpWInDow;

cmCur : Begin

IsResist:=False;

DisableCommAnds([cmCur]);

EnableCommAnds([cmRes]);

ShemeWInDow^.DrawView;

End;

cmRes : Begin

IsResist:=True;

DisableCommAnds([cmRes]);

EnableCommAnds([cmCur]);

ShemeWInDow^.DrawView;

End;

Else

Exit;

End;

ClearEvent(Event);

End;

End;

**Procedure *TMyApp.OpenSheme***;

Var

D: PFileDialog;

FileName: String[79];

i,j:Integer;

f:Text;

c:wOrd;

Begin

If Not Exist

Then NewSheme;

D:= PFileDialog(ValidView(New(PFileDialog, Init('\*.shm', 'Выбор файла',

'~И~мя файла со схемой', fDopenButton, 100))));

If D <> Nil Then

Begin

c:=Desktop^.ExecView(D);

If c <> cmCancel Then

Begin

D^.GetFileName(FileName);

Assign(f,FileName);

reset(f);

For i:=1 To nS Do Begin For j:=1 To mS Do Read (f,Sheme[i,j,1]); Readln(f);

End;

For i:=1 To nS Do Begin For j:=1 To mS Do Read(f,EDS[i,j]); Readln(f); End;

For i:=1 To nS Do Begin For j:=1 To mS Do Read(f,Res[i,j]); Readln(f); End;

Close(f); ShemeName:=FileName;

DisposeStr(ShemeWInDow^.Title);

ShemeWInDow^.Title:=NewStr('Схема '+ShemeName);

ElNumbers(Sheme);

ShemeWInDow^.DrawView;

End; Dispose(D, Done); End;

End;

**Procedure *TMyApp.SaveSheme***;

Var f:Text;

i,j:Integer;

Begin

Assign (f,ShemeName); ReWrite (f);

For i:=1 To nS Do Begin

For j:=1 To mS Do Write(f,Sheme[i,j,1]:4); Writeln(f);

End;

For i:=1 To nS Do Begin For j:=1 To mS Do Write(f,EDS[i,j]:5:2,' ');

Writeln(f);

End;

For i:=1 To nS Do Begin For j:=1 To mS Do Write(f,Res[i,j]:5:2,' ');

Writeln(f);

End;

Close(f);

End;

**Procedure *TMyApp.SaveShemeAs***;

Var

D: PFileDialog;

FileName: String[79];

W: PWInDow;

C:wOrd;

Begin

D:= New(PFileDialog, Init('\*.SHM', 'Выбор файла', ShemeName,

fDokButton, 100));

C:= Desktop^.ExecView(D); D^.GetFileName(ShemeName);

Dispose(D, Done);

If ShemeName='' Then exit; SaveSheme;

ShemeWInDow^.Title:=NewStr('Схема '+ShemeName);

ShemeWInDow^.DrawView;

End;

**Procedure *TMyApp.HlpWInDow***;

Var

W: PWInDow;

Begin

W:= PWInDow(ValidView(New(PFileWInDow,Init(HelpName))));

W^.HelpCtx:= hcMenu30; If W <> Nil Then Desktop^.Insert(W);

End;

**Procedure *TMyApp.NewSheme***;

Var R:TRect;

c:wOrd;

Begin

If Exist And Changed

Then Begin

R.Assign((Size.X-CurrentFont^.Width\*50) Div 2,(Size.Y-

CurrentFont^.Height\*10) Div 2,

(Size.X+CurrentFont^.Width\*50) Div 2,(Size.Y+

CurrentFont^.Height\*10) Div 2);

c:=MessageBoxRect(R,

'В текущую схему внесены изменения с момента последнего '+

'сохранения, которые будут потеряны. Хотите ли Вы сохранить текущую '+

'схему?',Nil,mfYesNoCancel+mfConfirmation);

Case c Of

cmYes:If ShemeName<>'' Then SaveSheme Else SaveShemeAs;

cmCancel:Exit; End; End;

If Not Exist Then Begin

DeskTop^.GeTextent(R);

ShemeWInDow:=New(PShemeWIn,Init(R));

DeskTop^.Insert(ShemeWInDow);End;

ShemeName:=''; DisposeStr(ShemeWInDow^.Title);

ShemeWInDow^.Title:=NewStr('Схема без имени');

Changed:=False; InitSheme(Sheme); ShemeWInDow^.DrawView;

End;

**Procedure *TMyApp.About***;

Begin

MessageBox('Обсчет разветвленных цепей. Ver.1.0.',Nil,

mfInFormation+mfOkButton);

End;

**Procedure *TMyApp.ReCounte***;

Begin Abstract;{This method must be overriden} End;

**END**.