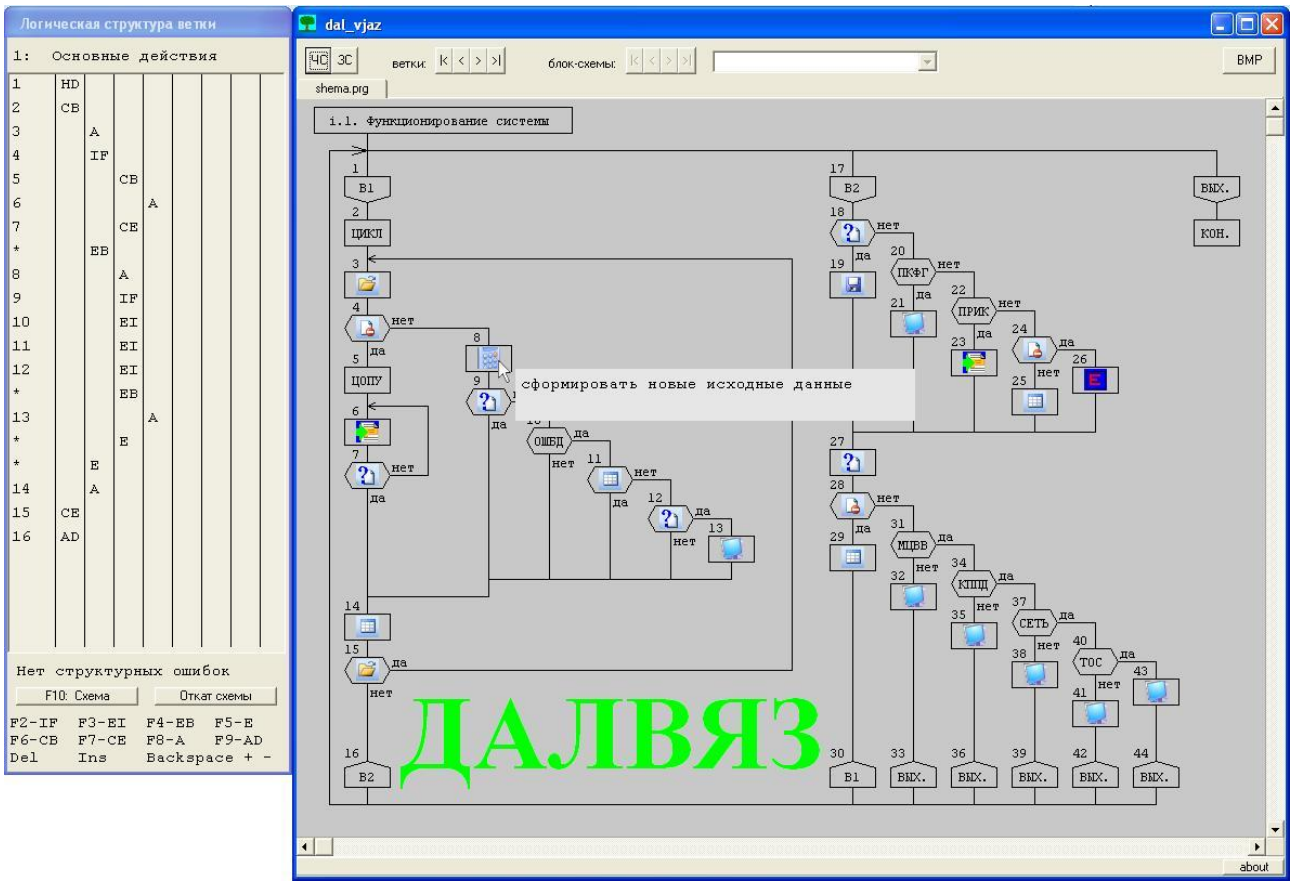
Д. Барановский



ДРУЖЕЛЮБНЫЙ АЛГОРИТМИЧЕСКИЙ ВИЗУАЛЬНЫЙ ЯЗЫК

(Подмножество языка ДРАКОН)

Интерфейс пользователя программы DAL_VJAZ

и

алгоритм создания визуальной схемы

Барановский Дмитрий Владимирович
ведущий программист
отдела горочной автоматизации
ОАО «НИИАС»

Предлагаемое читателям описание языка ДАЛВЯЗ (подмножества языка ДРАКОН) и программы DAL_VJAZ адресовано тем, кто интересуется вопросами визуального программирования. Алгоритмический язык ДРАКОН представляет одно из самостоятельных и развивающихся направлений в этой области. В этом описании интерфейс и алгоритм построения визуальной схемы рассматриваются «изнутри», с точки зрения программиста.

СОДЕРЖАНИЕ

ИСПОЛЬЗУЕМЫЕ СОКРАЩЕНИЯ

ДРАКОН	−	Дружелюбный Русский Алгоритмический язык, Который Обеспечивает Наглядность
ЯПВУ	−	язык программирования высокого уровня
CASE	−	система автоматизированного проектирования
ВС	−	визуальная схема
ЛСВ	−	логическая структура ветки ВС
ПкДАЛВЯЗ	−	псевдокод языка ДАЛВЯЗ
СТИ	−	структурно-текстовый интерфейс редактирования ВС

ВВЕДЕНИЕ

Данный материал возник как результат моего участия в обсуждении на форуме образовательного сайта oberoncore.ru в темах "дракон-си" и "Система разработки и документирования ПО РВ" путей и возможностей для дальнейшего развития систем визуального программирования, опирающихся на идеи, предложенные Владимиром Даниеловичем Паронджановым в процессе создания алгоритмического визуального языка ДРАКОН. В связи с чем выражаю свою благодарность и говорю большое СПАСИБО всем участникам обсуждения, которые своими высказываниями и предложениями заставили меня на многие вопросы посмотреть по-новому и прийти в конце концов к тем идеям, которые и легли в основу предлагаемого читателям описания языка ДАЛВЯЗ и программы dal_vjaz, на практике (хотя и в макетном варианте) реализующей работу с этим языком.

В своей первой программе, реализующей идеи В.Д. Паронджанова (программа ab_vjaz, тема "Система разработки и документирования ПО РВ", идея силуэта) я посчитал для себя идею о запрете на пересечение линий визуальной схемы слишком сложной. В программе dal_vjaz идея о запрете на пересечение линий визуальной схемы, а также идея о структурном текстовом представлении визуальной схемы реализованы с приложением сравнительно небольших усилий за счет отказа от досрочных выходов из циклов и запрета на непосредственное редактирование визуальной схемы пользователем. Для этого я решил написать визуальный интерфейс создания ВС, отличающийся от базового интерфейса графического редактора языка ДРАКОН, реализованного Г.Н. Тышовым в его программе и.с.Дракон. Интерфейс программы dal_vjaz для редактирования логической структуры ветки силуэта использует идеи, позаимствованные из интерфейса структурных редакторов. Фактически ЛСВ создается в структурно-текстовом формате и уже после проверки ее правильности визуализируется. Насколько такой интерфейс удобен – судить читателям.

Пока что программа dal_vjaz 0.2, выложенная в теме "дракон-си", представляет собой недоделанный до конца макет простого редактора алгоритмов ПО.

Вопрос: А зачем нужно читать про какой-то там недоделанный редактор алгоритмов ?

Ответ:

Многочисленные команды тестировщиков шерстят подготавливаемое к выпуску софтверными фирмами ПО и в хвост и в гриву, выискивая оставшихся в нем жуков. Естественно, что программист-одиночка лишен всех этих возможностей, а потому и качество его ПО будет заведомо уступать качеству фирменного ПО. Встречаются изредка отдельные уникумы, которые и в

одиночку ухитряются создавать достаточно сложное, качественное и стабильно работающее ПО, но эти исключения только подтверждают общее правило - программисты-одиночки находятся в заведомо проигрышном положении по отношению к софтверным фирмам.

Поэтому по такому сомнительному пути - в одиночку написать и хорошо задокументировать ПО полноценного и стабильно работающего редактора алгоритмов ПО - я решил не идти, а вместо этого решил взять для себя в качестве примера авторов книг по программированию, которые пишут: «Вы конечно понимаете, что здесь и здесь должна быть обработка ошибок, но для упрощения понимания основной логики работы программы мы ее вставлять не будем, потому что лучше хорошо описать работу небольшой учебной программки, чем плохо и поверхностно описать работу большой и сложной программы». Вот и я займусь подробным описанием моего простенького и недоделанного макета.

И алгоритмы, используемые моей программой, далеки от оптимальных, и чтение визуальной схемы из файла с последующей проверкой выполняется долго (несколько секунд), и вид самой визуальной схемы даже в ее минимизированном варианте далек от совершенства. В общем много шероховатостей, да и неотлаженные ошибки скорее всего имеются (куда уж без них!).

Но я и не стремился достигнуть совершенства - моей задачей было создать макет-заготовку, которую энтузиасты визуального программирования, если бы мой макет показался им интересным, могли бы всячески развивать и дорабатывать как в смысле скорости и надежности работы программы, так и в смысле улучшения эргономичности отображения визуальной схемы и пользовательского интерфейса.

Для понимания излагаемого материала, относящегося к программированию, читателю потребуется знание основ языка C++ и знакомство с IDE Borland C++ Builder 3.0 (или выше). В исходном коде процедур и методов, реализующих алгоритм создания визуальной схемы, ООП используется главным образом для взаимодействия с VCL. Рассматриваемый алгоритм построения визуальной схемы читатель сможет, если захочет, без особых затруднений перенести на любой другой из объектно-ориентированных или процедурных ЯПВУ.

Алгоритмический язык ДРАКОН

ДРАКОН (Дружелюбный Русский Алгоритмический язык, Который Обеспечивает Наглядность) - визуальный алгоритмический язык, созданный в рамках космической программы Буран. С помощью этого визуального языка стало возможным программировать сложные системы не программистам, а специалистам в своей предметной области. Разработка данного языка была начата в 1986 г. под руководством Владимира Даниеловича Пароджанова. В разработке языка принимали участие Российское космическое агентство (НПЦ автоматики и приборостроения, г. Москва) и Российская академия наук (Институт прикладной математики им. М.В. Келдыша, г. Москва).

На базе языка ДРАКОН построена автоматизированная технология проектирования программных систем (CASE-технология) под названием "ГРАФИТ-ФЛОКС". На выходе − ПО на проблемно-ориентированном языке высокого уровня для некоей специализированной управляющей ЭВМ (например, бортовой ЭВМ ракетного комплекса). Эта технология успешно используется в ряде крупных космических проектов: "Морской старт", "Фрегат", "Протон-М" и др.

Работы по разработке языка окончательно были завершены в 1998 году (спустя 5 лет после закрытия программы Буран), когда была создана CASE-технология Графит-Флокс.

В.Д. Пароджанов в своей книге "Как улучшить работу ума" указывает на возможность трансляции ДРАКОН-схем непосредственно в программный код на языках высокого уровня − Си, Паскале и т.п..

Можно выделить следующие основные идеи языка ДРАКОН:

1) Главным понятием языка ДРАКОН является «силуэт» (см. Рис. 1).

Силуэт позволяет разбить вытянутую по вертикали блок-схему с беспорядочными связями между блоками на несколько взаимосвязанных логических фрагментов (веток), расположенных в силуэте по горизонтали слева направо, ссылающихся друг на друга по адресам веток и графически замыкающихся друг на друга при помощи обратной петли силуэта, расположенной слева от самой левой ветки силуэта. Как написал на форуме oberoncore И.Е. Ермаков, один из преподавателей программирования ОГТУ, силуэт является средством переключения состояний и в одном из пониманий визуальным отображением конечного автомата.

2) С целью улучшения читаемости визуальных схем алгоритмов В.Д. Пароджанов постулировал запрет на пересечение линий визуальной схемы (единственное исключение – точка пересечения обратной петли силуэта с главной вертикалью визуальной схемы), хотя ГОСТ 19.701-90, описывающий правила выполнения схем алгоритмов, этого не требует.

3) Ввод визуальной схемы выполняется в специализированном графическом редакторе путем помещения визуальных элементов в нужные участки ВС с последующим заданием их размеров и текстового содержания.

4) При вводе схемы графический редактор выполняет жесткую проверку на «структурность» - блокируются все анархические переходы типа перехода извне внутрь цикла и т.п. (подробное описание правил ввода визуальных схем в языке ДРАКОН см. в книге В.Д. Пароджанова «Как улучшить работу ума»).

Из-за того, что я не располагаю вышеперечисленными производственными ресурсами, для написания моего сравнительно простого редактора алгоритмов ПО мне пришлось ограничиться упрощенной реализацией некоторого минимального подмножества языка ДРАКОН, достаточного для генерации программ на ЯПВУ.

Т.к. названия «недоДРАКОН» и «ДРАКОН--» выглядят некрасиво, то я решил дать используемому мной подмножеству этого языка свое собственное название, расшифровка которого по смыслу почти идентична расшифровке сокращения ДРАКОН, данной В.Д. Пароджановым.

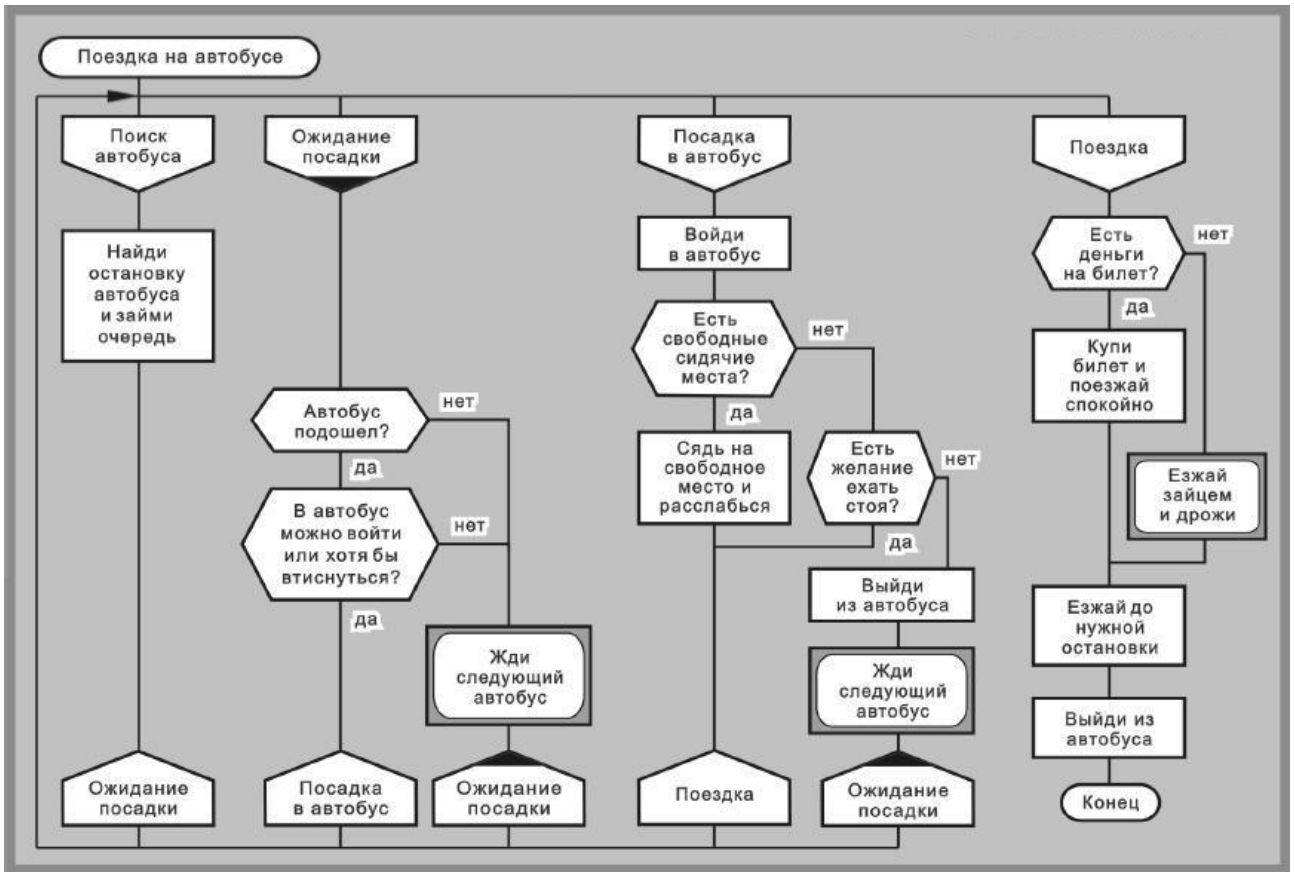


Рис. 1. Полная форма отображения визуальной схемы языка ДРАКОН

ДАЛВЯЗ

Алгоритмический язык ДАЛВЯЗ предназначен для создания и отображения на мониторе компьютера минимизированных визуальных схем подмножества языка ДРАКОН, являющихся графической формой изображения логики процедур исходного текста программы.

Вопрос: Как понимать фразу "минимизированная визуальная схема" ?

Ответ:

Полная форма отображения визуальной схемы языка ДРАКОН изображена на Рис. 1, а минимизированная форма отображения визуальной схемы (или, если короче, минимизированная визуальная схема) - на Рис. 2.

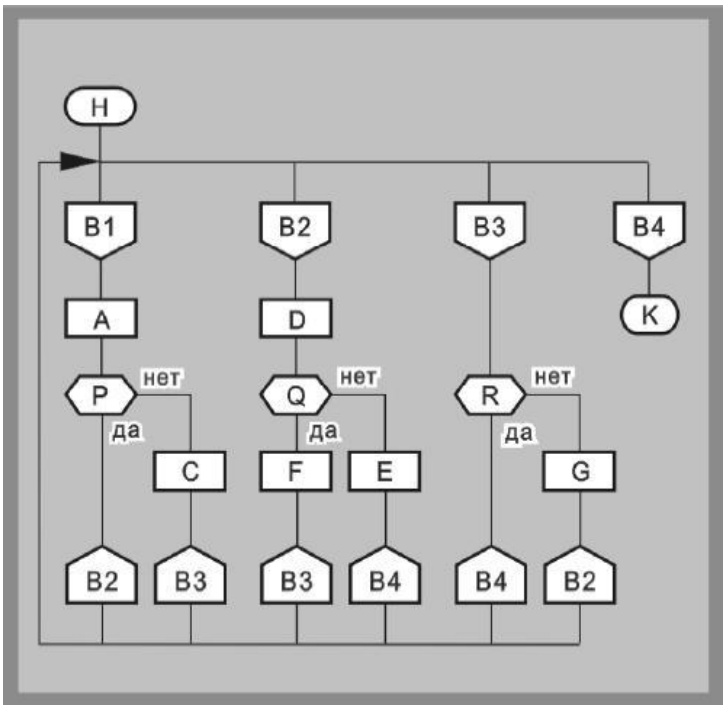


Рис. 2. Минимизированная форма отображения визуальной схемы

В ДАЛВЯЗе название элемента визуальной формы кодируется сокращением максимум из 4-х букв или вообще может быть заменено на пиктограмму, а комментарии и код исходного текста программы для элемента появляются на экране при наведении указателя мыши на этот элемент.

Такая форма отображения визуальной схемы на экране монитора сокращает информационное давление на программиста и позволяет разместить на экране больший фрагмент визуальной схемы, чем это позволяет сделать полная форма отображения визуальной схемы (при том же размере шрифта).

Изменения, введенные в ДАЛВЯЗе по сравнению с языком ДРАКОН

В языке ДАЛВЯЗ используется только визуальная схема «силуэт», поэтому понятия «силуэт» и «визуальная схема» в ДАЛВЯЗе равнозначны.

В ДАЛВЯЗе обрабатываются блоки визуальной схемы следующих типов:
- «заголовок»;
- «начало ветки»;
- «действие»;
- «условие»;
- «начало цикла»;
- «конец цикла» (условие с переходом вверх к началу цикла);
- «адрес ветки»;
- «КОНЕЦ».

В ДАЛВЯЗе используется только минимизированная форма визуальной схемы, хотя возможен ее экспорт в полную форму отображения визуальной схемы.

Для обозначения элемента внутрь его блока могут быть помещены, по выбору, или сокращенное название элемента, или пиктограмма, обозначающая действие, выполняемое элементом. Причем одна и та же пиктограмма может иметь смысл как утверждения, так и вопроса в зависимости от того, относится ли она к блоку «действие» или к блоку «условие».

Всем визуальным элементам ВС (кроме заголовка схемы, а также заголовка и оператора «КОНЕЦ» ветки «ВЫХОД») в обязательном порядке присваиваются порядковые номера – это является необходимым требованием используемого в программе dal_vjaz структурно-текстового интерфейса ввода ВС.

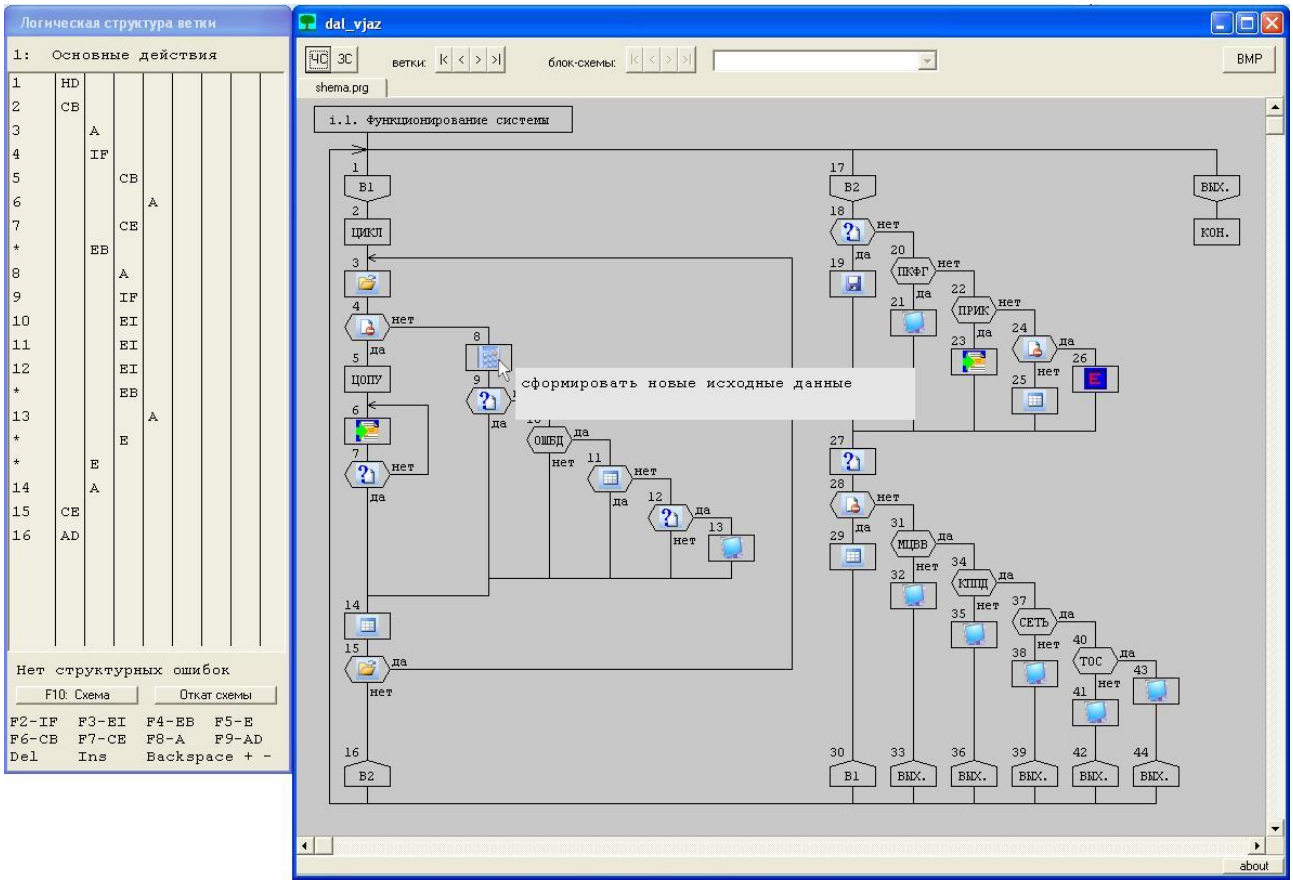
Являясь алгоритмическим визуальным языком, ДАЛВЯЗ включает в себя не только подмножество набора визуальных элементов и правил построения визуальных схем языка ДРАКОН, но и свой собственный псевдокод для описания структурного текстового представления визуальной схемы.

Визуальный интерфейс программы dal_vjaz

Вид главного окна программы dal_vjaz представлен на Рис. 3.

Главное окно программы dal_vjaz состоит из окна визуальной схемы и окна логической структуры ветки ВС, выбранной пользователем.

Наводя указатель мыши на элементы визуальной схемы пользователь может во всплывающем окне элемента прочитать информацию об исходном коде и (или) комментарии этого элемента. После щелчка правой кнопкой мыши на элементе на экране появляется контекстное меню элемента (см. Рис. 4).



1. Основные действия
2. основной цикл функционирования системы
3. считывание исходных данных
4. проверка корректности исходных данных
5. цикл опроса устройств
6. опрос устройств
7. данные считаны ?
8. сформировать новые исходные данные
9. исходные данные сформированы ?
10. ошибка в базе данных
11. проверка целостности базы данных
12. нужен вывод диагностики ?
13. вывод диагностики состояния системы
14. обработка данных
15. провести новое считывание исходных данных ?
17. обработка диагностических сообщений
18. проверка полученных данных
19. сохранить полученные данные
20. проверка корректности конфигурации
21. конфигурация задана корректно, но полученные данные неверны
22. получение нового файла конфигурации ?
23. прием нового файла конфигурации
24. фатальная ошибка данных ?
25. восстановление данных
26. сообщение о фатальной ошибке данных
27. проверка правильности функционирования системы
28. система функционирует правильно ?
29. обработка данных для возвращения к основным действиям
31. проверка модулей цифрового ввода/вывода
32. вывод диагностики о состоянии системы
34. проверка каналов последовательной передачи данных
35. вывод диагностики о состоянии системы
37. проверка сетевых продключений
38. вывод диагностики о состоянии системы
40. тестирование оборудования системы
41. вывод диагностики о состоянии системы
43. вывод диагностики о состоянии системы

Рис. 3. Вид главного окна программы dal_vjaz с пояснениями для элементов ВС

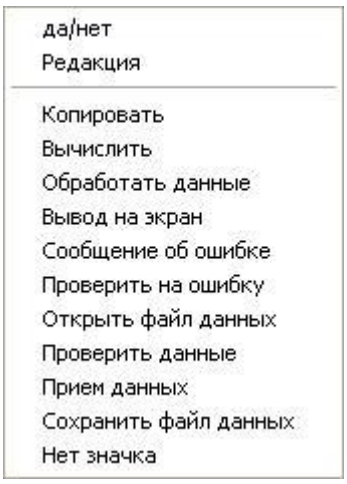


Рис. 4. Контекстное меню элемента ВС

Контекстное меню элемента ВС позволяет пользователю редактировать комментарии и текст исходного кода программы для элемента, менять местами названия веток условий (да/нет), а также задавать нужный значок для элемента. Пиктограммы элементов программы dal_vjaz представлены на Рис. 5. Этот набор пиктограмм выбран для примера. А если говорить о разработке реальных систем, то каждая группа разработчиков должна сама определять тот набор рабочих сокращений и пиктограмм, которым она будет пользоваться при разработке своей системы. Понятно, что для обозначения неких обобщенных действий удобнее использовать уже общепринятые пиктограммы, если для этих действий такие пиктограммы имеются (например, операции копирования данных, открытие и сохранение файлов и т.п.).

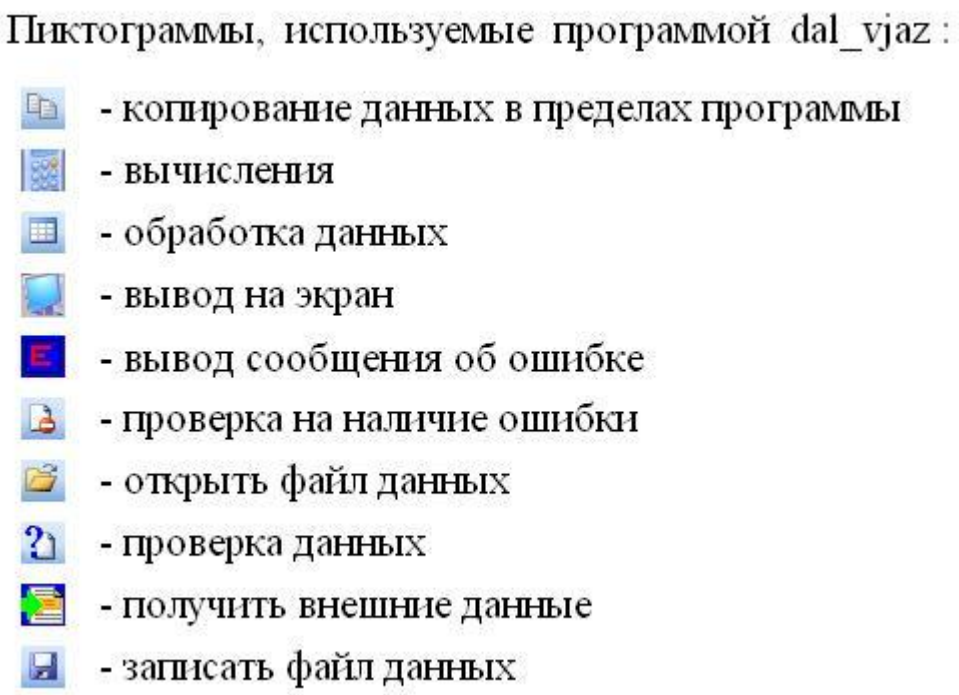


Рис. 5. Пиктограммы элементов ВС программы dal_vjaz

В случае выбора в контекстном меню элемента строки «Редакция» на экране появляется окно редактирования элемента, в котором, отделенные друг от друга символами "-" будут находиться поля редактирования данных элемента:

сокращение элемента или имя bmp-файла пиктограммы

ссылка на другую ВС или на html-файл

поле комментариев элемента

поле текста исходного кода для элемента

Если одно из полей элемента к моменту окончания редактирования останется незаполненным, то оно должно состоять хотя бы из одной пустой строки.

В окне логической структуры ветки пользователь может задавать и изменять логическую структуру текущей ветки ВС (работа с окном логической структуры ветки подробно рассматривается ниже). Пользователь может выбирать текущую ветку ВС как в окне визуальной схемы, щелкая мышью по кнопкам « |< », « < », « > », « >| », так и в окне ЛСВ, нажимая клавиши «+» и «-» цифровой группы клавиш клавиатуры.

Пользователь может сохранить отредактированную ВС, щелкнув мышью по кнопке «ЗС» - запись схемы. После этого в каталоге программы dal_vjaz будут сохранены новые версии файлов shema.prg (файл структурного текстового представления ВС) и shema.txt (файл комментариев к элементам ВС; на Рис. 3 пояснения для элементов ВС взяты из этого файла).

Пользователь может считать в программу dal_vjaz новую ВС, для чего нужно заменить файлы shema.prg и shema.txt (наличие этого файла необязательно) в каталоге программы dal_vjaz, а затем щелкнуть мышью по кнопке «ЧС» - чтение схемы.

Структурное текстовое представление визуальной схемы

Псевдокод ДАЛВЯЗа (ПкДАЛВЯЗ) для описания структурного текстового представления визуальной схемы включает в себя минимальный набор операторов, необходимый для создания полноценного исходного кода программы. Все операторы ПкДАЛВЯЗ однозначно соответствуют аналогичным операторам условий и циклов современных объектно-ориентированных и процедурных ЯПВУ, а потому и текст программы на

ПкДАЛВЯЗ может быть легко и однозначно переведен в текст программы на любом из этих языков. Для справки в нижеследующем описании операторов ПкДАЛВЯЗ приводятся и соответствующие им операторы языка Паскаль.

ПкДАЛВЯЗ включает в себя следующие операторы:

1) Оператор начала блока условия:

IF <+|-> //.<номер элемента> <сокращение|файл пиктограммы>
строка условия 1
...
строка условия N

Начало блока по выполнению (+) или невыполнению (-) условия

Обозначение оператора IF в ЛСВ: IF
Обозначение типа оператора IF в программе dal_vjaz: t_IF

Начало блока условия в языке Паскаль: if (usl) then begin

2) Оператор конца блока условия:

END

Обозначение оператора END в ЛСВ: E
Обозначение типа оператора END в программе dal_vjaz: t_E

Конец блока условия в языке Паскаль: end;

3) Оператор альтернативного блока условия:

ELSE

Обозначение оператора ELSE в ЛСВ: EB
Обозначение типа оператора ELSE в программе dal_vjaz: t_EB

Альтернативный блок условия в языке Паскаль: end else begin

4) Оператор дополнительного блока условия:

ELSE IF <+|-> //.<номер элемента> <сокращение|файл пиктограммы>
строка условия 1
...
строка условия N

Начало блока по выполнению (+) или невыполнению (-) условия

Обозначение оператора ELSE IF в ЛСВ: EI
Обозначение типа оператора ELSE IF в программе dal_vjaz: t_EI

Дополнительный блок условия в языке Паскаль: end else if (usl) then begin

5. Оператор начала цикла

CB //.<номер элемента> <сокращение>

Обозначение оператора CB в ЛСВ: CB
Обозначение типа оператора CB в программе dal_vjaz: t_CB

Начало цикла в языке Паскаль: repeat

6. Оператор конца цикла

CE <+|-> //.<номер элемента> <сокращение|файл пиктограммы>
строка условия 1
...
строка условия N

Выход из цикла по выполнению (+) или невыполнению (-) условия

Обозначение оператора CE в ЛСВ: CE
Обозначение типа оператора CE в программе dal_vjaz: t_CE

Окончание цикла в языке Паскаль: until (usl);

7. Оператор заголовка ветки BC

HD <номер ветки> //.<номер элемента>
строка 1 описания ветки
...
строка N описания ветки

Обозначение оператора HD в ЛСВ: HD
Обозначение типа оператора HD в программе dal_vjaz: t_HD

Начало ветки BC в языке Паскаль: отсутствует, см. пояснение ниже

8. Оператор адреса перехода к другой ветке BC

AD <номер ветки|EXIT> //.<номер элемента>

Обозначение оператора AD в ЛСВ: AD
Обозначение типа оператора AD в программе dal_vjaz: t_AD

Переход к другой ветке ВС в языке Паскаль: отсутствует, см. пояснение ниже

9. Оператор действия

строка 1 действия //.<номер элемента> <сокращение|файл пиктограммы>
...
строка N действия

Обозначение оператора действия (action) в ЛСВ: A
Обозначение типа оператора действия в программе dal_vjaz: t_A

Оператор действия в языке Паскаль: любой исходный код на языке Паскаль,
например: `a := b+c;`

10. Оператор окончания процедуры

RETURN

Обозначение оператора RETURN в ЛСВ: отсутствует
Обозначение типа оператора действия в программе dal_vjaz: t_RET

Окончание процедуры в языке Паскаль: end;

Всем операторам ПкДАЛВЯЗ (кроме ELSE и END) соответствуют признаки начала элемента ВС вида:

//.<номер элемента> <сокращение|файл пиктограммы>

Номер элемента может отсутствовать.

<сокращение|файл пиктограммы> (может отсутствовать):

XXXX - четырехсимвольное сокращение действия/условия элемента

или

<имя bmp-файла> - имя файла, содержащего пиктограмму элемента, которая будет помещена внутрь блока элемента

В простейшем случае признак начала элемента имеет вид: //.

Вопрос: Зачем нужны признаки начала элементов ?

Ответ: Тут 2 причины:
1) В элемент действия могут входить несколько операторов программы и только программист знает общую логику процедуры и понимает, после какого оператора логически заканчивается одно действие и начинается другое.
2) В текстовом файле shema.txt содержатся строки комментариев к процедуре и их удобно привязывать к отдельным логическим действиям процедуры при помощи индексов.

Пояснения для операторов HD и AD ПкДАЛВЯЗ

В языке Паскаль этим операторам будут соответствовать языковые конструкции цикла, условия и присвоения нового значения переменной. Например, если переменная силуэта имеет имя usl_vjaz, то получим:

```
begin  { начало процедуры }

  usl_vjaz := 1;

  while (usl_vjaz <> 0) do begin   { цикл силуэта }

        if (usl_vjaz = 1) then begin  { 1-я ветка силуэта }

              действия ветки 1 силуэта

              usl_vjaz := N;  { переход к ветке N силуэта }
        end;

        ...

        if (usl_vjaz = N) then begin  { N-я ветка силуэта }

              действия ветки N силуэта

              usl_vjaz := 0;  { на выход }
        end;
  end;
end;  { конец процедуры }
```

Пример текста процедуры на ПкДАЛВЯЗ

В качестве примера привожу текст файла shema.prg, соответствующего ВС, показанной на Рис. 3:

```
// i.1. Функционирование системы

     HD  1                                                    //.1
       CB                                                     //.2  ЦИКЛ
                                                              //.3  p_otkr.bmp
         IF  +                                                //.4  p_iserr.bmp
           CB                                                 //.5  ЦОПУ
                                                              //.6  p_recdt.bmp
           CE  +                                              //.7  p_prowdt.bmp
         ELSE
                                                              //.8  p_culc.bmp
           IF  +                                              //.9  p_prowdt.bmp
           ELSE IF  -                                         //.10  ОШБД
           ELSE IF  +                                         //.11  p_data.bmp
           ELSE IF  -                                         //.12  p_prowdt.bmp
           ELSE
                                                              //.13  p_ekran.bmp
           END
         END
                                                              //.14  p_data.bmp
       CE  -                                                  //.15  p_otkr.bmp
       AD  2                                                  //.16
     HD  2                                                    //.17
       IF  +                                                  //.18  p_prowdt.bmp
                                                              //.19  p_zakr.bmp
       ELSE IF  +                                             //.20  ПКФГ
                                                              //.21  p_ekran.bmp
       ELSE IF  +                                             //.22  ПРИК
                                                              //.23  p_recdt.bmp
       ELSE IF  -                                             //.24  p_iserr.bmp
                                                              //.25  p_data.bmp
       ELSE
                                                              //.26  p_err.bmp
       END
                                                              //.27  p_prowdt.bmp
       IF  +                                                  //.28  p_iserr.bmp
                                                              //.29  p_data.bmp
         AD  1                                                //.30
       ELSE IF  -                                             //.31  МЦВВ
                                                              //.32  p_ekran.bmp
         AD  EXIT                                             //.33
       ELSE IF  -                                             //.34  КППД
                                                              //.35  p_ekran.bmp
         AD  EXIT                                             //.36
       ELSE IF  -                                             //.37  СЕТЬ
                                                              //.38  p_ekran.bmp
         AD  EXIT                                             //.39
       ELSE IF  -                                             //.40  ТОС
                                                              //.41  p_ekran.bmp
         AD  EXIT                                             //.42
       ELSE
                                                              //.43  p_ekran.bmp
         AD  EXIT                                             //.44
       END
     HD  EXIT    //.
     RETURN      //.
```

Видно, что у элементов t_A вышеприведенной процедуры заданы только признаки начала элементов, а их смысловая часть осталась незаполненной.

Первые итоги знакомства с dal_vjaz: чего нет и что должно быть

Вот так выглядит программа dal_vjaz на текущий момент. Сразу видно, что в ней многого не хватает. Сейчас перечислю самое основное:

1) Нет возможности работать с конкретными ЯПВУ.
При желании можно исправить исходный текст программы dal_vjaz так, чтобы она могла сохранять и считывать визуальную схему не на псевдокоде, а на заданном в файле конфигурации dal_vjaz конкретном ЯПВУ (да и сам файл конфигурации не помешает, а то его сейчас нет).

2) Нет работы с файлами ЯПВУ.
Тут в качестве примера можно взять за основу мысли, реализованные в программе ab 1.40, тема "дракон-си".

Логика состоит в том, чтобы рассматривать каждый файл исходного кода как набор записей, состоящих из заголовка, огражденного символами комментария (см. заголовок вышеприведенного файла shema.prg), и собственно текста исходного кода программы, находящегося в теле записи.

Текст на ЯПВУ, обрабатываемый dal_vjaz, должен быть помещен внутрь блока текста визуальной схемы:

```
// i. заголовок записи

начало процедуры

//.begin

текст визуальной схемы

//.end

окончание процедуры
```

При считывании файла ЯПВУ во внутренний формат dal_vjaz кроме записей элементов мы получим и запись заголовка процедуры вида:

начало процедуры

//.begin
//.end

окончание процедуры

которую можно будет редактировать в dal_vjaz как обычный текстовый файл. То же справедливо и для записей, в состав которых не входит блок текста визуальной схемы.

3) Нет реализации ссылки одной схемы на другую.
Здесь в качестве примера можно взять программу ab_vjaz из темы "Система разработки и документирования ПО РВ".

4) Нет работы с ветками визуальной схемы.
Как минимум нужны следующие операции:

- вставка новой ветки; ветка вставляется сразу вместе с заголовком ветки; заголовок ветки может быть удален только вместе со своей веткой;
- удаление существующей ветки;
- обмен 2-х веток местами, при котором программа автоматически должна отслеживать изменения адресов переходов к другим веткам.
- откат схемы к состоянию до выполнения операции над ветками (реализован частично и на макетном уровне – надо бы улучшить).

Возможно я и забыл перечислить что-нибудь, но эти четыре пункта можно считать необходимыми требованиями по дальнейшему развитию программы dal_vjaz.

А пока вернемся к обсуждению работы текущей версии программы dal_vjaz.

Редактирование текущей ветки визуальной схемы в окне ЛСВ

Начнем с того, что присвоим файлам shema.prg и shema.txt в каталоге программы dal_vjaz какие-нибудь другие имена, а затем запустим программу dal_vjaz. Когда программа обнаружит, что визуальной схемы нет, она создаст минимальную визуальную схему по умолчанию, показанную на Рис. 6.

Добавим к этой схеме 2-ю ветку. Для этого откроем shema.prg в текстовом редакторе и скопируем строки 4 и 5 в строки 6 и 7, а в строке 6 исправим 1 на 2. После этого сохраним изменения shema.prg, перейдем в

dal_vjaz и щелкнем мышью по кнопке «ЧС». Считанная схема показана на Рис.7.

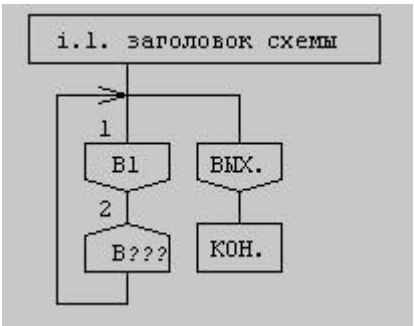


Рис. 6. Минимальная ВС, создаваемая dal_vjaz при отсутствии файла shema.prg

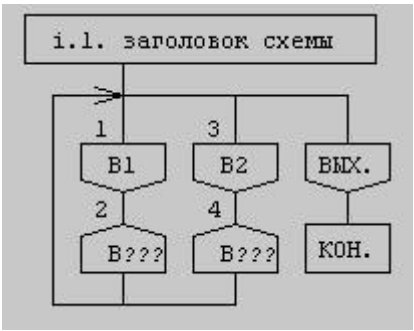


Рис. 7. Полученная ВС с двумя ветками

После этого выберем мышью строку 2 окна ЛСВ и последовательно будем нажимать на следующие функциональные клавиши клавиатуры: «F2», «F8», «F4», «F8», «F5», после чего список операторов в окне ЛСВ для ветки 1 примет следующий вид:

```
1       HD
        IF
                A
        EB
                A
        E
2       AD
```

После этого нажмем клавишу «F10», и схема примет вид, показанный на Рис.8.

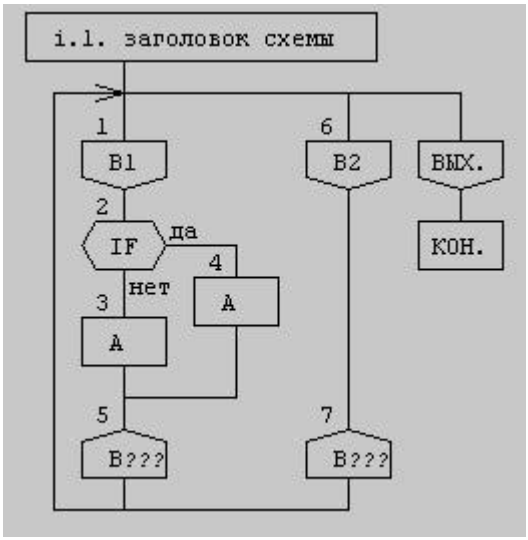


Рис. 8. ВС после добавления блока IF - ELSE

А теперь перенесем добавленный блок IF – ELSE в ветку 2. Для этого выделим мышью или с клавиатуры (удерживая Shift и нажимая «↓») строки 2, 3, 4, 5 и 6 списка ЛСВ, чтобы невыделенными остались только HD в начале и AD в конце списка. Затем нажмем клавишу Delete, и выделенные строки будут удалены в буфер ЛСВ. После этого нажмем «F10», а затем перейдем к ветке 2, нажав клавишу «+» цифровой группы клавиатуры. ВС на экране сдвинется так, чтобы ее 2-я ветка оказалась рядом с левым краем окна ВС, а в окне ЛСВ появится логическая структура ветки 2, состоящая из операторов HD и AD. Поместим выделенную строку списка ЛСВ на оператор AD и нажмем клавишу Insert, а затем «F10», а затем вернемся к ветке 1, нажав клавишу «-»цифровой группы клавиатуры. После этого окно ЛСВ примет вид, показанный на Рис. 10.

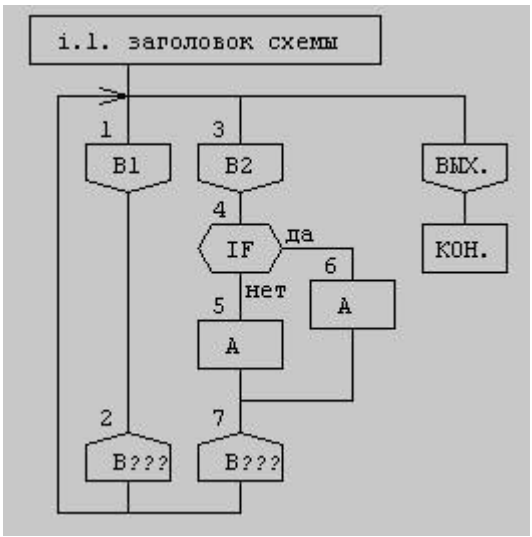


Рис. 10. ВС после переноса блока IF – ELSE в ветку 2

Вот примерно так и выполняется редактирование ВС в программе dal_vjaz.

Кстати, во время переноса блока IF из ветки 1 в ветку 2 можно было наблюдать одну из шероховатостей – не фатальная ошибка конечно, но это говорит о том, что с алгоритмом вывода ВС надо еще работать.

Когда блок IF был удален из ветки 1, а ВС после этого была обновлена по «F10», она приняла вид, показанный на Рис.11.

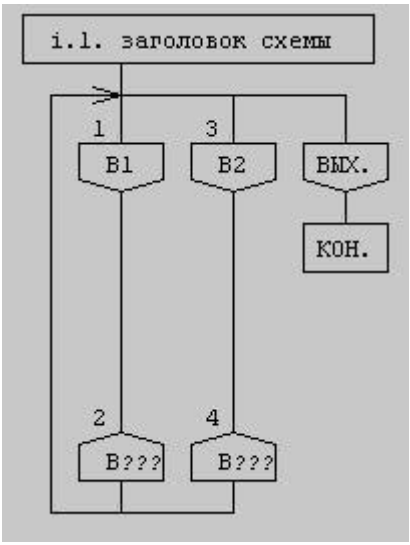


Рис. 11. А координаты элементов после удаления IF пересчитать забыли

Это мелочь – чтобы заставить программу выполнить пересчет координат, нужно сначала щелкнуть мышью по кнопке «ЗС», а затем по кнопке «ЧС».

В качестве еще одного примера редактирования ВС создадим визуальную схему досрочного выхода из цикла. Пользуясь предыдущим опытом, создадим ВС, показанную на Рис. 12.

ЛСВ для ветки 1 будет следующим:

```
1       HD
2       IF
3               A
4               A
*       EB
5               A
*       E
6       AD
```

Установим выделенную строку ЛСВ на элемент «3 A» и нажмем на клавиатуре «F6», а затем на «* EB» и нажмем «F7». После этого нажмем «F10».

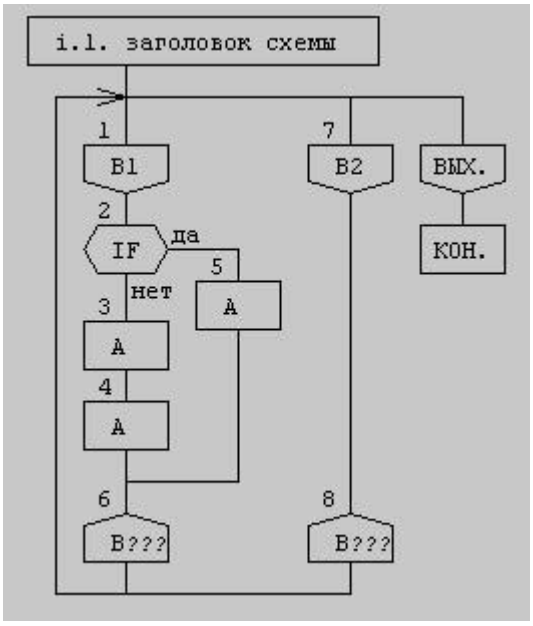


Рис. 12. Заготовка для ВС досрочного выхода из цикла

Таким образом мы задали цикл, охватывающий оба действия основной ветки IF.

Теперь ЛСВ будет выглядеть так:

```
1       HD
2       IF
3               CB
4                       A
5                       A
6               CE
*       EB
7               A
*       E
8       AD
```

Чтобы сформировать досрочный выход из цикла, наводим выделенную строку ЛСВ на «5 A», нажимаем «F2», затем наводим выделенную строку на «6 CE» и нажимаем «F4», «F8», «F5», «F10». Затем отредактируем действие A, которое получит номер 7 и будет лежать на ветке «да» условия внутри цикла. Установим для этого действия сокращение ВЫХЦ – выход из цикла (предполагается, что в исходном коде этого действия мы взведем флаг выхода из цикла, который будет проверять оператор окончания цикла). Получившаяся ВС изображена на Рис. 13.

Конечно, сначала структурно-текстовый интерфейс (СТИ) ввода ВС кажется слегка непривычным, но по моему мнению скорость построения ВС для СТИ как минимум не меньше скорости построения ВС в программе и.с.Дракон Г.Н. Тышова.

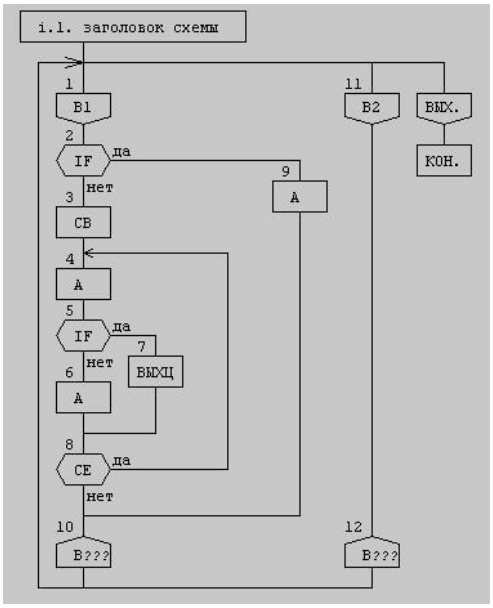


Рис. 13. ВС досрочного выхода из цикла

Системы координат программы dal_vjaz

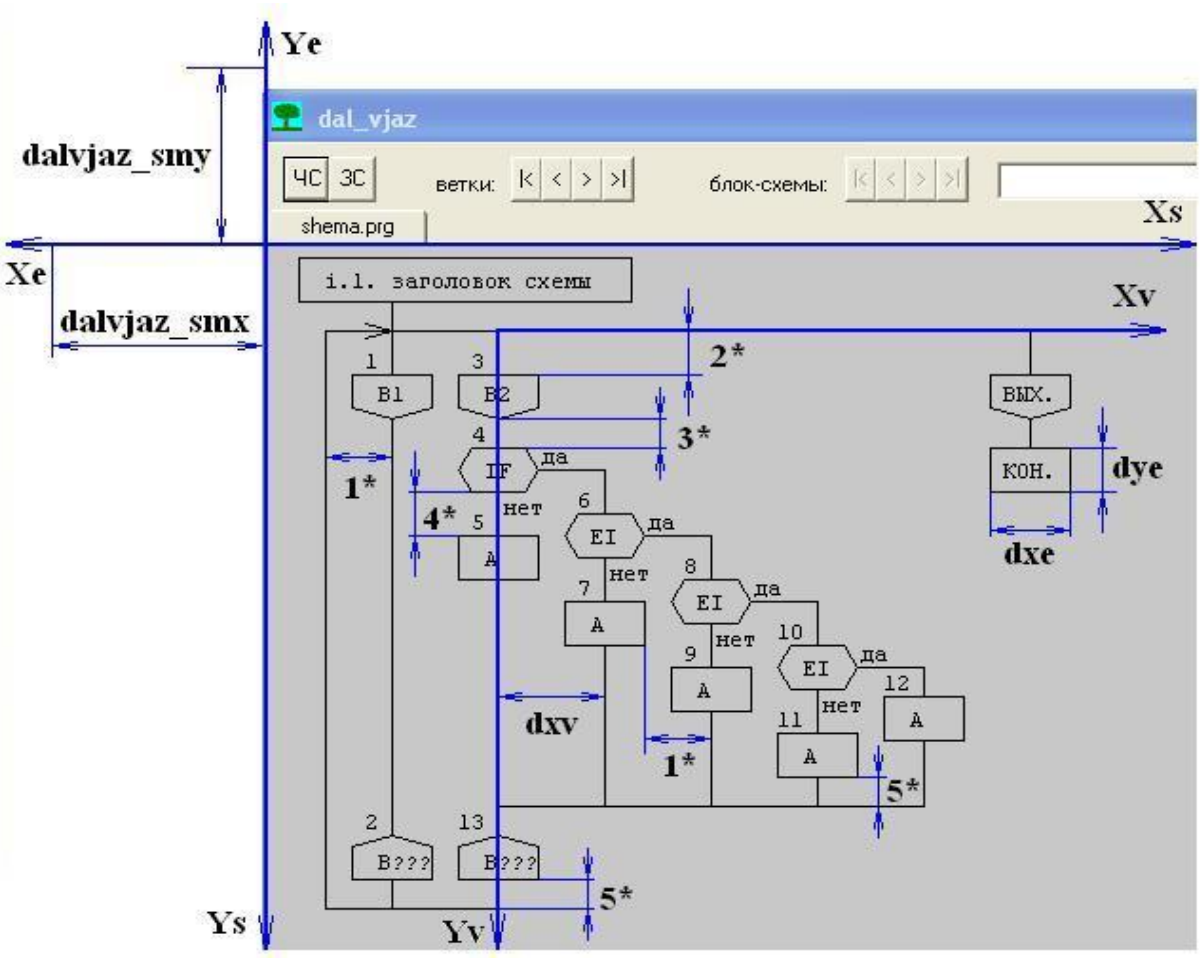


Рис. 14. Системы координат программы dal_vjaz

В программе dal_vjaz используются 3 системы координат:
1) Система координат окна ВС Xe, Ye.
2) Система координат визуальной схемы Xs, Ys.
3) Система координат вертикали Xv, Yv.

Начала и направления осей координат для всех вышеперечисленных систем координат программы dal_vjaz показаны на Рис. 14.

Оси системы координат окна ВС Xe, Ye направлены в противоположную сторону по отношению к осям системы координат Xs, Ys потому, что при продвижении бегунков полос прокрутки окна ВС вправо и вниз увеличиваются значения переменных dalvjaz_smx и dalvjaz_smy, а сама схема смещается влево и вверх, уходя за левую и верхнюю границы окна ВС.

Система координат вертикали Xv, Yv используется для задания координат элементов, лежащих на этой вертикали.

Как видно из Рис. 14, схема состоит из основных вертикалей, вершины которых лежат на верхней горизонтали силуэта, и порождаемых лежащими на главной вертикали ветки элементами «условие» и «начало цикла» дочерних вертикалей и вертикалей цикла (вертикаль цикла тоже является дочерней, но ее надо было назвать по-другому, чем дочернюю вертикаль, порождаемую условием).

Для простоты реализации я принял, что расстояния между всеми вертикалями одинаковы и равны dxv.

Кроме того, габаритные размеры всех элементов также одинаковы и равны dxe и dye.

На Рис. 14 также показаны следующие размеры:

- 1* - расстояние между вертикалью и правой границей лежащей слева от нее вертикали (для самой левой вертикали схемы – расстояние между вертикалью и обратной петлей силуэта);
- 2* - расстояние от верхней горизонтали силуэта до первого элемента вертикали;
- 3* - расстояние между элементом «заголовок» или «действие» и следующим элементом вертикали;
- 4* - расстояние между элементом типа «условие», «начало цикла» или «конец цикла» и следующим элементом вертикали;
- 5* - расстояние между последним элементом ветки и горизонталью, ограничивающей эту ветку снизу.

Я решил, что удобно было бы измерять расстояния по x и по y на схеме в размерах символов моноширинного фонта, используемого программой dal_vjaz для вывода текста элементов ВС. За шаг по x я принял ширину символа dalvjaz_chw, а за шаг по y – расстояние по y между строками выводимого текста dalvjaz_chdy.

Тогда вышеуказанные размеры будут равны:

dxv = 8* dalvjaz_chw
dxe = 6* dalvjaz_chw
dye = (3* dalvjaz_chdy)/2
1* = 5* dalvjaz_chw
2* = (3* dalvjaz_chdy)/2
3* = dalvjaz_chdy
4* = (3* dalvjaz_chdy)/2
5* = dalvaz_chdy для элемента «адрес ветки»; не меньше dalvaz_chdy для элемента «действие»; не меньше (3* dalvjaz_chdy)/2 для элементов «условие» и конец цикла.

Кстати, для вывода текста схемы программа dal_vjaz использует фонт «Courier New», 9 пт., а разрешение монитора у меня стоит 1280*1024.

Основные типы данных и переменные программы dal_vjaz

Основные типы данных и переменные программы dal_vjaz определены в заголовочном файле U_LS.H (см. Приложение А).

В состав основных типов входят:

t_wert - тип для вертикали ВС;
t_elem - тип для элемента ВС;
t_alg - тип для текущей ВС;
el_lsw - тип для элемента ЛСВ;
el_stotm - тип для элемента стека отмены ввода ЛСВ.

В состав основных переменных входят:

t_alg tek_alg - текущая ВС;
el_lsw m_lsw[RAZM_LSW] - массив элементов ЛСВ;
int kol_m_lsw - текущее число элементов массива m_lsw;
el_stotm stotm[RAZM_STOTM_LSW] - стек отмены ввода ЛСВ; в окне ЛСВ отмена ввода выполняется по «Backspace»;
int kol_stotm - текущее число элементов в stotm;

el_lsw buf_lsw[RAZM_BUF_LSW] - буфер копирования ЛСВ (клавишами «Delete» и «Insert»);
int kol_b_lsw - текущее число элементов в buf_lsw;
el_lsw stek_lsw[RAZM_STEKA_LSW] - стек ЛСВ, используется при построении ВС;

int kol_s_lsw - текущее число элементов в stek_lsw;
el_lsw tek_el_shemy - текущий элемент при построении схемы;
el_lsw el_steka_lsw - переменная для чтения эл-та из стека ЛСВ;
int m_otkr_wert[KOL_OTKR_WERT] - массив открытых и одновременно создаваемых вертикалей;
int kol_otkr_wert - число одновременно открытых вертикалей;
int tek_otkr_wert - текущая открытая вертикаль;
int dalvjaz_chw - ширина символа текущего фонта;
int dalvjaz_chh - высота символа текущего фонта;
int dalvjaz_chdy - шаг по у между двумя строками;
int dalvjaz_smx - смещение ВС по х;
int dalvjaz_smy - смещение ВС по у;
int dalvjaz_xmax - максимальная х для схемы;
int dalvjaz_ymax - максимальная у для схемы;
int dalvjaz_tek_wetka - текущая ветка ВС.

Пояснения по работе переменной tek_alg

Перед созданием ветки для текущих элементов ветки удаляются ссылки на вертикали ветки, а затем удаляются все вертикали ветки.
После окончания создания ветки будут удалены те элементы ветки, которые остались не привязанными к вертикалям (поле old типа t_elem у этих элементов после построения схемы будет равно IS).

Т.к. элементы удаляются и вставляются хаотически (путем сброса и взведения флага used), то нельзя поручиться, что и элементы одной ветки будут следовать друг за другом в массиве элементов по порядку их следования в ЛСВ. Поэтому для заново построенной ветки:
1) находим индексы элементов ветки в массиве элементов;
2) сортируем элементы ветки по порядку их следования в ЛСВ (поле n_str типа t_elem).

В то время как основные вертикали схемы расположены в массиве tek_alg.m_wert начиная с нулевого индекса, дочерние вертикали по мере их создания при построении схемы размещаются в массиве tek_alg.m_wert начиная с индекса KOL_WETOK.

Правила построения визуальной схемы

Ниже приводятся не связанные явным образом между собой правила построения ВС, которые лежат в основе алгоритма построения визуальной схемы и реализованы в методах и процедурах программы dal_vjaz, выполняющих построение ВС.

- При считывании ВС из файла и до момента создания схемы элементы ВС приписываются только к веткам, их n_wert = -1.

- Ветка ВЫХОД записывается последней и будет отличаться от других веток только тем, что ее нельзя будет удалять и редактировать.

- Расстояние по y от начала координат схемы до верхней горизонтали ВС:

  DALVJAZ_Y_ZAGOL + ((5*dalvjaz_chdy)/2)

- Для ветки после считывания файла ВС известны m_el_wetki, m_inewet.

- У главной вертикали ветки n_sprawa = 0. Главная вертикаль ветки может быть удалена только вместе с веткой, а дочерние вертикали удаляются перед построением отредактированной или считанной ВС и в процессе построения ВС создаются заново.

- Массив открытых вертикалей m_otkr_wert содержит индексы открытых вертикалей. В начале формирования ветки у него только 1 элемент – главная вертикаль ветки, а индекс текущей открытой вертикали tek_otkr_wert = 0.

- В начале создания вертикали задаются ее тип (дочерняя вертикаль или вертикаль цикла), N ветки, tyel, y1 в координатах схемы, m_el_wert = 0, m_inew = -1.

- Новая вертикаль вставляется в массив открытых вертикалей после своего родителя и ее n_sprawa будет равна n_sprawa родителя + 1.

- 1-й элемент главной вертикали ветки будет ниже верхней горизонтали ВС на (3*dalvjaz_chdy)/2, это значение будет присвоено tyel главной вертикали ветки.

- Когда создается t_IF:

1) создается новый элемент;
2) для элемента создается новая вертикаль и привязывается к нему; tek_otkr_wert сохраняется.

- Когда создается t_EI:

1) находим t_IF или предыдущий t_EI и делаем его вертикаль текущей еще перед созданием t_EI;
2) создаем элемент на новой вертикали;
3) для созданного элемента в свою очередь создается новая вертикаль, аналогично t_IF.

- Когда создается t_EB:

1) ищем в стеке ЛСВ t_IF или t_EI и делаем его вертикаль текущей;
2) t_EB создается на этой ветрикали;
3) y1 и y2 для t_EB равны 0;
4) в стек ЛСВ t_EB не заносится, в отличие от t_IF и t_EI.

- Когда создается t_E:

1) делаем текущей родительскую вертикаль t_IF;
2) создаем t_E;
3) у t_E y1 = y2, а расстояние по y до следующего элемента такое же, как у действия;
4) создаем массив открытых вертикалей блока t_IF, включая родительскую ветрикаль t_IF и для этих вертикалей ищем ту, чья tyel находится ниже других. Эту координату берем для задания нижних у всех вертикалей этого массива.
5) все другие вертикали созданного t_E массива вертикалей, кроме вертикали t_IF, удаляем из массива открытых вертикалей.

- t_CB создает новую вертикаль, а открытой остается текущая вертикаль.

- t_CE закрывает открытую t_CB вертикаль, задавая для нее tyel. Число элементов вертикали цикла равно 0.

- Для t_A и t_AD просто добавляем элемент на текущую откытую вертикаль.

- dalvjaz_ymax находим как максимальный tyel для главных вертикалей.

- Вывод вертикалей на схеме:

1) вывод главных вертикалей веток;
2) вывод дочерних вертикалей веток.

- Для главных вертикалей рисуем линии от верхнего элемента вертикали до верхней горизонтали BC.

- Для последней главной вертикали (ВЫХ.) от последнего элемента не рисуем линию вниз.

- Для дочерних вертикалей рисуем линии соединения с родительскими вертикалями.

- Размер по x ветки считаем по максимальной n_sprawa ее дочерних вертикалей.

Перечень и описание методов и процедур модулей U_LS и U_DALVJ

Для удобства я проиндексировал методы и процедуры программы dal_vjaz и теперь могу искать их в файлах текста программы не по именам, а по индексам, а ссылки на методы/процедуры можно указывать как
<имя модуля>.<индекс метода/процедуры>

Главными методами программы dal_vjaz являются:

- u_dalvj.29: b_chtenieClick - чтение ВС из файла;
- u_dalvj.21: wywod_shemy - вывод схемы;
- u_ls.9: b_sozdatx_shemuClick - создать из списка ЛСВ ветку визуальной схемы;
- u_dalvj.31: b_zapisxClick - запись ВС в файл.

Остальные методы и процедуры или выполняют достаточно простые действия по обработке запросов VCL-интерфейса форм ЛСВ и ВС, или в логическом смысле являются обработчиками служебных подзадач, возникающих в процессе работы вышеуказанных основных методов программы dal_vjaz.

```
модуль  u_ls
============

// i.2. конструктор формы ЛСВ

__fastcall Tform_ls::Tform_ls(TComponent* Owner)


// i.3. действия при создании формы ЛСВ

void __fastcall Tform_ls::FormCreate(TObject *Sender)


// i.4. самостоятельное рисование строки списка ЛСВ как строки таблицы

void __fastcall Tform_ls::ListBox1DrawItem(TWinControl *Control, int Index,
      TRect &Rect, TOwnerDrawState State)


// i.5. обработка клавиатуры для списка ЛСВ
//      клавиши:  F2 - F8, F10, Backspace,
//      Delete, Insert, Ctrl+X, Ctrl+C, Ctrl+V,
//      "+" цифровой группы, "-" цифровой группы

void __fastcall Tform_ls::ListBox1KeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)


// i.6. начальная инициализация после создания формы ВС
//      синхронизация положения с окном ВС
```

```
void __fastcall Tform_ls::Timer1Timer(TObject *Sender)

// i.7.

// i.8.

// i.9. создать из списка ЛСВ ветку визуальной схемы

void __fastcall Tform_ls::b_sozdatx_shemuClick(TObject *Sender)

          * начало
             * помечаем 1-ю выделенную строку списка ЛСВ

             * устанавливаем параметры главной вертикали текущей ветки

             * инициализация массива открытых вертикалей ветки

             * запомнить состояние схемы до начала обработки для
               возможного отката

             * текущая строка списка ЛСВ = 0

             * номер ошибки сброшен

             * удаляем вертикали для текущей ветки

             * если номер ошибки != 0, переход к обработке ошибки

             * сброс стека ЛСВ

             * текущая строка списка ЛСВ = 0

             * цикл сканирования ЛСВ

                * задание текущей строки ЛСВ

                * получить текущий элемент схемы из ЛСВ

                * получить вершину стека ЛСВ

                * проверка ошибок логической структуры ветки

                * если ошибка, то переход к обработке ошибок

                * если (тип == t_IF или t_EI или t_CB) то начать

                   * если  t_IF  то  обработка для t_IF

                   * если  t_EI  то  обработка для t_EI

                   * если  t_CB  то  обработка для t_CB

                   * если ошибка, то переход к обработке ошибок

                   * заносим текущий элемент в стек ЛСВ
                * конец если
                * иначе

                * если  t_EB  то начать

                   * обработка для  t_EB
```

```
      * если ошибка, то переход к обработке ошибок
    * конец если
    * иначе

    * если  t_E  то начать

      * обработка для  t_E

      * если ошибка, то переход к обработке ошибок

      * цикл

        * если в стеке ЛСВ нет элементов, то
          переход к обработке ошибок

        * извлечь элемент из стека ЛСВ

        * если извлечен  t_IF, то выход из цикла
      * конец цикла
    * конец если
    * иначе

    * если  t_CE  то начать

      * обработка для  t_CE

      * если ошибка, то переход к обработке ошибок

      * если в стеке ЛСВ нет элементов, то
        переход к обработке ошибок

      * извлечь элемент из стека ЛСВ
    * конец если
    * иначе начать

      * обработка для  t_HD, t_A, t_AD

      * если ошибка, то переход к обработке ошибок
    * конец иначе

* конец цикла сканирования ЛСВ

* получить вершину стека ЛСВ

* если в стеке ЛСВ остались элементы, то
  переход к обработке ошибок

* обработка после конца сканирования ЛСВ

* если номер ошибки сброшен, то начать
   * если схема считана, то начать

      * // здесь надо будет вставить пересчет координат
        // элементов схемы после ее обработки

      * вывод схемы

      * передать фокус ввода окну ЛСВ

      * загрузка ЛСВ

      * вернуть выделение той строке ЛСВ, которая была выделена
        перед началом создания схемы
```

```
            * конец если
          * конец если

          * обработка ошибок

        * конец


// i.10. удалить вертикали текущей ветки

void __fastcall Tform_ls::udal_wert_tek_wetki(TObject *Sender)


// i.11. обработка ошибок

void __fastcall Tform_ls::obrabotka_oshibok(TObject *Sender)


// i.12. проверить ошибки структуры

void __fastcall Tform_ls::prow_oshibki_struktury(TObject *Sender)


// i.14. обработка схемы для  t_IF

void __fastcall Tform_ls::obr_shemy_dlq_IF(TObject *Sender)

        * обработать схему для элемента

        * если ошибка, то выход

        * создать дочернюю вертикаль


// i.15. создать вертикаль

void __fastcall Tform_ls::sozdatx_wert(int type)

            Создание новой дочерней вертикали
          и заполнение массива открытых вертикалей

        Дочерняя вертикаль всегда создается для текущей открытой вертикали.

        Новая вертикаль всегда ставится на место с n_sprawa на 1 больше,
        чем у эл. слева, а ранее открытые вертикали сдвигаются вправо и при
        необходимости их n_sprawa последовалельно увеличиваются на 1 слева
        направо так, чтобы у каждой открытой вертикали n_sprawa оставался
        уникальным.
        При закрытии вертикали и ее удалении из массива открытых вертикалей
        элементы массива открытых вертикалей, находящиеся правее, сдвигаются
        влево.
        К моменту закрытия вертикали ее n_sprawa определяется окончательно,
        так что становится возможным найти ее смещение относительно главной
        вертикали ветки по X. В момент закрытия вертикали элементом t_E
        находим и нижнюю координату У для этой вертикали.


// i.16. обработка схемы для  t_EI

void __fastcall Tform_ls::obr_shemy_dlq_EI(TObject *Sender)

        *  ищем  дочернюю вертикаль для находящегося в стеке ЛСВ t_IF или
           t_EI в массиве открытых вертикалей и делаем ее текущей
```

```
          * обработка для элемента

          * создать дочернюю вертикаль


// i.17. обработка схемы для  t_EB

void __fastcall Tform_ls::obr_shemy_dlq_EB(TObject *Sender)

          *  ищем  дочернюю вертикаль для находящегося в стеке ЛСВ t_IF или
             t_EI в массиве открытых вертикалей и делаем ее текущей

          * обработка для элемента


// i.18. обработка для  t_E

void __fastcall Tform_ls::obr_shemy_dlq_E(TObject *Sender)


         t_E составляет список всех открытых IF и EI вертикалей и закрывает их


         // ветку t_IF делаем текущей открытой веткой

         // ищем в стеке t_IF

         // загружаем во врем.мас.откр.верт. все вертикали t_IF

         // для всех эл. врем.мас.откр.верт. проверяем, а есть ли
         // найденный элемент в массиве открытых вертикалей

         // вычисляем максимальную y для всех найденных вертикалей

         // для всех вертикалей задаем новое значение tyel, задаем ссылку
         // на t_E и удаляем их из массива открытых вертикалей
         // вертикаль t_IF остается открытой

         // корректируем координату y для t_E


// i.19. обработка для  t_CB

void __fastcall Tform_ls::obr_shemy_dlq_CB(TObject *Sender)

         * обработка схемы для элемента

         * создать вертикаль цикла


// i.20. обработка для  t_CE

void __fastcall Tform_ls::obr_shemy_dlq_CE(TObject *Sender)

         // исключаем вертикаль цикла из массива открытых вертикалей
         // задаем ей нижнюю границу и элемент конца


// i.21. обработка для  t_HD, t_A, t_AD

void __fastcall Tform_ls::obr_shemu_dlq_HD_A_AD(TObject *Sender)

         * обработка схемы для элемента
```

```
// i.22. обработка схемы для элемента

void __fastcall Tform_ls::obr_shemu_dlq_el(TObject *Sender)

         // если новый элемент силуэта, то для него нужно выделить
         // место в массиве элементов

         // если к эл. списка ЛСВ элемент еще не привязан, то
         // поиск в массиве ЛСВ элемента, соответствующего
         // формально месту в массиве элементов схемы

         // нинциализация элемента схемы

         // находим координаты эл-та на вертикали
         // ELSE помещаем на верхнюю вершину порожденной вертикали

         // находим tyel для вертикали после задания координат эл-та


// i.23. функция сортировки "меньше" для массива элементов

int less_m_el(int i, int j)


// i.24. функция сортировки "поменять местами" для массива элементов

void  swap_m_el(int i, int j)


// i.25. обработка после окончания сканирования ЛСВ

void __fastcall Tform_ls::obr_shemy_w_konce_lsw(TObject *Sender)

         // удаление элементов текущей ветки, которые остались старыми
         // не привязанными к вертикалям

         // новое число элементов схемы

         // находим индексы элементов ветки

         // сортируем элементы ветки по порядку следования в ЛСВ

         // новый индекс начального элемента ветки

         // находим для дополнительных вертикалей ветки m_inew

         // восстанавливаем для вертикалей ветки ссылки на родительские
         // элементы и на элементы конца ветки

         // ищем элементы с нужными для родительского элемента
         // вертикали и конца вертикали номерами веток и строк ЛСВ

         // присваиваем элементам схемы новые порядковые номера


// i.26. запомнить состояние схемы
//       записываем файлы отката

void __fastcall Tform_ls::zapomnitx_sostoqnie(TObject *Sender)


// i.27. вернуть состояние схемы
//       читаем файлы отката
```

```
void __fastcall Tform_ls::wernutx_sostoqnie(TObject *Sender)

// i.28.

// i.29. обработка для кнопки "откат схемы"
void __fastcall Tform_ls::b_otmena_shemyClick(TObject *Sender)

// i.30. запись ЛСВ из схемы
void __fastcall Tform_ls::zapisx_LSW_iz_shemy(TObject *Sender)

// i.31. чтение ЛСВ
void __fastcall Tform_ls::chtenie_LSW(TObject *Sender)

// i.32. проверка буфера ЛСВ при замене ветки
void  prow_buf_lsw_pri_smene_wetki()
         // не должно быть ссылок из буфера на активный эл. схемы

// i.33. загрузка ЛСВ
void __fastcall Tform_ls::zagruzka_LSW(TObject *Sender)

// i.34. поместить элемент в стек ЛСВ
void  push_stek_lsw()

// i.35. получить элемент из стека ЛСВ
void  pop_stek_lsw()

// i.36. получить запрашиваемый элемент стека ЛСВ
void  give_stek_lsw(int n)

// i.37. сброс стека ЛСВ
void  sbros_steka_lsw()

// i.38. действия при активации формы ЛСВ
void __fastcall Tform_ls::FormActivate(TObject *Sender)

// i.39. действия при деактивации формы ЛСВ
void __fastcall Tform_ls::FormDeactivate(TObject *Sender)
```

```
// i.40. обработка щелчка мыши по форме ЛСВ

void __fastcall Tform_ls::FormClick(TObject *Sender)


// i.41. обработка щелчка мыши по списку ЛСВ

void __fastcall Tform_ls::ListBox1Click(TObject *Sender)


// i.42. инициализация массива открытых вертикалей

void  mow__init(int tw)


// i.43. получить для запрашиваемой открытой вертикали
//       индекс в массиве вертикалей схемы

int  mow__n_wert(int n)


// i.44. вставить верт. с индексом tw на n-е место
//       массива открытых вертикалей

void  mow__wstawitx(int tw, int n)


// i.45 удалить вертикаль из массива открытых вертикалей

void  mow_udalitx(int n)


// i.46 преобразовать тип элемента в строку

void  type_el_to_str(int type, char *stro)


// i.47. проверка является ли строка пустой или строкой пробелов

uchar  str_probelow(char *str)


// i.48. заготовка для пересчета координат элементов после
//       завершения построения схемы

void __fastcall Tform_ls::pereschet_koord_el(TObject *Sender)



модуль  u_dalvj
===============


// i.2. вывод линии для ВС

void  dalvjaz_line(int x1, int y1, int x2, int y2)


// i.3. вывод текста для ВС

void  dalvjaz_str(int x, int y, char* stro)
```

```
// i.4. задание размера фонта для ВС

void  dalvjaz_font(int n)

// i.5. конструктор формы ВС

__fastcall Tform_dalvjaz::Tform_dalvjaz(TComponent* Owner)

// i.6. обработка для создания формы ВС

void __fastcall Tform_dalvjaz::FormCreate(TObject *Sender)

         * считываем начальные размеры окна ВС и пиктограммы

// i.7. обработка для вертикальной полосы прокрутки ВС

void __fastcall Tform_dalvjaz::ScrollBar1Change(TObject *Sender)

// i.8. обработка для изменения размеров формы ВС

void __fastcall Tform_dalvjaz::FormResize(TObject *Sender)

// i.9. обработка для кнопки "о программе"

void __fastcall Tform_dalvjaz::b_aboutClick(TObject *Sender)

// i.10. обработка перерисовки ВС

void __fastcall Tform_dalvjaz::PaintBox1Paint(TObject *Sender)

// i.11. обработка при удалении формы

void __fastcall Tform_dalvjaz::FormDestroy(TObject *Sender)

// i.12. обработка при щелчке мыши по ВС

void __fastcall Tform_dalvjaz::PaintBox1MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)

         // обработка для правой кнопки мыши - вывод контекстного меню
         // если щелкнули по элементу

         // сначала проверяем элементы основных вертикалей веток,
         // а затем элементы дочерних вертикалей веток

// i.14. обработка для кнопки BMP - создание файла shema.bmp

void __fastcall Tform_dalvjaz::b_bmpClick(TObject *Sender)

// i.15. обработка для горизонтальной полосы прокрутки ВС

void __fastcall Tform_dalvjaz::ScrollBar2Change(TObject *Sender)
```

```
// i.16. обработка кнопки "к первой ветке"

void __fastcall Tform_dalvjaz::b_w_nClick(TObject *Sender)


// i.17. обработка кнопки "к последней ветке"

void __fastcall Tform_dalvjaz::b_w_kClick(TObject *Sender)


// i.18. обработка кнопки "к предыдущей ветке"

void __fastcall Tform_dalvjaz::b_w_minClick(TObject *Sender)


// i.19. обработка кнопки "к следующей ветке"

void __fastcall Tform_dalvjaz::b_w_plClick(TObject *Sender)


// i.20. обработка движения мыши по ВС

void __fastcall Tform_dalvjaz::PaintBox1MouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)



// i.21. вывод схемы

void __fastcall Tform_dalvjaz::wywod_shemy(TObject *Sender)

         // задание фонта цвета окна (серый или белый для bmp)

         // рисуем заголовок

         // если элементов нет или не проверены все
         // ветки схемы, то выход

         // подсчитываем размер схемы

         // вертикальная линия от заголовка до верхней горизонтальной
         // линии силуэта

         // рисуем верхнюю горизонтальную линию силуэта

         // стрелка обратной петли силуэта

         // рисуем все вертикали
         // сначала выводим элементы основных вертикалей веток, а затем
         // элементы дополнительных вертикалей веток

         // нижняя горизонтальная линия силуэта

         // вертикальная линия обратной петли силуэта


// i.22. вывод элемента схемы

void __fastcall Tform_dalvjaz::wywod_elem(TObject *Sender, int w, int ie, int n)

         // рамка элемента

         // текст элемента
```

```
          // вывод пиктограммы

          // индекс элемента

          // ДА/НЕТ

          // рисуем линию к следующему элементу
          // для последнего эл. метки ВЫХОД не нужно

          // проводим линию до следующего эл-та или, если эл. последний на
          // вертикали, то до  tyel  вертикали



// i.23. подсчитываем размер схемы

void __fastcall Tform_dalvjaz::podschet_koordinat(TObject *Sender)

          // считаем число дочерних вертикалей

          // подсчет  m_dx_wetok[tw], tek_alg.m_wert[w].x  в зависимости
          // от наличия дополнительных вертикалей

          // m_dx_wetok

          // для всех веток найдем максимальное смещение от главной
          // вертикали

          // m_x_wetok

          // смещения дочерних вертикалей

          // размер схемы по X

          // размер схемы по У

          // для всех главных вертикалей, если t_E последний, то
          // опускаем t_E на уровень нижней горизонтали схемы и то же с
          // tyel всех вертикалей, для которых этот t_E закрывающий

                // для текущего эл. ищем все дочерние вертикали, для
                // которых он является закрывающим, и корректируем у
                // них tyel

          // для всех вертикалей, у которых tyel упирается в нижнюю
          // горизонталь и t_AD является последним элементом, лежащим
          // выше t_E на нижней горизонтали, опускаем t_AD к нижней
          // горизонтали


// i.24. обработка закрытия формы ВС

void __fastcall Tform_dalvjaz::FormClose(TObject *Sender,
      TCloseAction &Action)


// i.25. удалить заданное число начальных пробелов в строке и,
//       если требуется, оставить между словами строки только 1 пробел

void  udal_probely(char *str, int nach, uchar fl_1_prob)


// i.26. формирование элемента при чтении схемы из файла
```

```
void  dalvjaz_form_el(int type, int te, int tw, int *ti, char *_up)

// i.27.

// i.28. закончить формирование элемента
void  zakonchitx_wwod_el(int te)

// i.29. чтение ВС из файла
void __fastcall Tform_dalvjaz::b_chtenieClick(TObject *Sender)

// i.30. задать признак элемента для строки при записи схемы в файл
void  zadatx_metku_stroki(int ii, char *stro)
// i.31. запись ВС в файл
void __fastcall Tform_dalvjaz::b_zapisxClick(TObject *Sender)

// i.32. обработка для всплывающего информационного окна элемента
int  wsplyw_inf_okno(int X, int Y, int j, int ii)

// i.33. обработка строки "да/нет" контекстного меню элемента
void __fastcall Tform_dalvjaz::mn_danetClick(TObject *Sender)

// i.34. обработка строки "редакция" контекстного меню элемента
void __fastcall Tform_dalvjaz::mn_redClick(TObject *Sender)

// i.35. обработка для значка "копирование данных"
void __fastcall Tform_dalvjaz::mn_copyClick(TObject *Sender)

// i.36. обработка для значка "вычислить"
void __fastcall Tform_dalvjaz::mn_calcClick(TObject *Sender)

// i.37. обработка для значка "обработка данных"
void __fastcall Tform_dalvjaz::mn_dataClick(TObject *Sender)

// i.38. обработка для значка "вывод на экран"
void __fastcall Tform_dalvjaz::mn_ekranClick(TObject *Sender)

// i.39. обработка для значка "сообщение об ошибке"
void __fastcall Tform_dalvjaz::mn_errClick(TObject *Sender)
```

```
// i.40. обработка для значка "проверка на ошибку"
void __fastcall Tform_dalvjaz::mn_iserrClick(TObject *Sender)

// i.41. обработка для значка "открыть файл данных"
void __fastcall Tform_dalvjaz::mn_otkrClick(TObject *Sender)

// i.42. обработка для значка "проверить данные"
void __fastcall Tform_dalvjaz::mn_prowdtClick(TObject *Sender)

// i.43. обработка для значка "получить внешние данные"
void __fastcall Tform_dalvjaz::mn_recdtClick(TObject *Sender)

// i.44. обработка для значка "сохранить файл данных"
void __fastcall Tform_dalvjaz::mn_zakrClick(TObject *Sender)

// i.45. отмена значка для элемента
void __fastcall Tform_dalvjaz::mn_net_znachkaClick(TObject *Sender)

// i.46 читать файл комментариев для элементов ВС
void __fastcall Tform_dalvjaz::read_comment(char *fn)

// i.47 записать файл комментариев для элементов ВС
void __fastcall Tform_dalvjaz::write_comment(char *fn)
```

ПРИЛОЖЕНИЕ А. U_LS.H – основной заголовочный файл dal_vjaz

```
//---------------------------------------------------------------------------
#ifndef u_lsH
#define u_lsH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class Tform_ls : public TForm
{
__published:      // IDE-managed Components
    TListBox *ListBox1;
    TLabel *Label1;
    TLabel *Label2;
    TLabel *Label3;
    TTimer *Timer1;
    TPanel *Panel1;
    TImage *Image1;
    TLabel *l_n_wetki;
    TLabel *l_z_n_wetki;
    TLabel *l_soob;
    TLabel *l_soob_oshibka;
    TButton *b_sozdatx_shemu;
    TButton *b_otmena_shemy;
    TLabel *l_soob_no_osh;
    void __fastcall FormCreate(TObject *Sender);
    void __fastcall ListBox1DrawItem(TWinControl *Control, int Index,
          TRect &Rect, TOwnerDrawState State);
    void __fastcall ListBox1KeyDown(TObject *Sender, WORD &Key,
          TShiftState Shift);
    void __fastcall Timer1Timer(TObject *Sender);
    void __fastcall FormMouseMove(TObject *Sender, TShiftState Shift,
          int X, int Y);

    void __fastcall b_sozdatx_shemuClick(TObject *Sender);
    void __fastcall b_otmena_shemyClick(TObject *Sender);


    void __fastcall FormActivate(TObject *Sender);
    void __fastcall FormDeactivate(TObject *Sender);
    void __fastcall FormClick(TObject *Sender);
    void __fastcall ListBox1Click(TObject *Sender);

private:    // User declarations
public:           // User declarations
    __fastcall Tform_ls(TComponent* Owner);

    int lsw_selected;    // индекс и число выделенных строк списка
    int lsw_kol_sel;
    int tek_str_lsw;     // текущая строка ЛСВ при создании схемы, начиная с 0
    int osh_sozd_shemy;  // номер ошибки создания схемы

    void __fastcall udal_wert_tek_wetki(TObject *Sender);
    void __fastcall obrabotka_oshibok(TObject *Sender);
```

```
    void __fastcall prow_oshibki_struktury(TObject *Sender);
    void __fastcall obr_shemy_dlq_IF(TObject *Sender);
    void __fastcall sozdatx_wert(int type);
    void __fastcall obr_shemy_dlq_EI(TObject *Sender);
    void __fastcall obr_shemy_dlq_EB(TObject *Sender);
    void __fastcall obr_shemy_dlq_E(TObject *Sender);
    void __fastcall obr_shemy_dlq_CB(TObject *Sender);
    void __fastcall obr_shemy_dlq_CE(TObject *Sender);
    void __fastcall obr_shemu_dlq_HD_A_AD(TObject *Sender);
    void __fastcall obr_shemu_dlq_el(TObject *Sender);
    void __fastcall obr_shemy_w_konce_lsw(TObject *Sender);
    void __fastcall zapomnitx_sostoqnie(TObject *Sender);
    void __fastcall wernutx_sostoqnie(TObject *Sender);
    void __fastcall zapisx_LSW_iz_shemy(TObject *Sender);
    void __fastcall chtenie_LSW(TObject *Sender);
    void __fastcall zagruzka_LSW(TObject *Sender);

};
//---------------------------------------------------------------------------
extern PACKAGE Tform_ls *form_ls;

// i.01. константы и типы

#define   k_F2          113
#define   k_F3          114
#define   k_F4          115
#define   k_F5          116
#define   k_F6          117
#define   k_F7          118
#define   k_F8          119
#define   k_F9          120
#define   k_F10         121
#define   k_Ins         45
#define   k_Del         46
#define   k_cifr_PLUS   107
#define   k_cifr_MIN    109
#define   k_Backspace   8

#define   uchar         unsigned char
#define   IS            1
#define   NO            0


// типы элементов силуэта

#define  t_IF     1  // если (условие), то начать блок
#define  t_EI     2  // конец блока иначе если (условие), то начать блок
#define  t_EB     3  // иначе начать блок
#define  t_E      4  // конец блока
#define  t_CB     5  // начало цикла
#define  t_CE     6  // конец цикла
#define  t_HD     7  // заголовок ветки силуэта
#define  t_AD     8  // адрес следующей ветки силуэта  N | EXIT
                     // N - номер следующей ветки
                     // EXIT - переход к ветке ВЫХОД
#define  t_A      9  // действие (action)
#define  t_RET   10  // конец, возврат из процедуры (RETURN)




#define  dl_stro          210
#define  rname_el         20   // размер имени элемента
#define  KOL_FAJLOW       10   // обработка до 10 файлов исходного кода
```

```
#define  RAZM_OTMENA      10   // число действий по редактированию схемы,
                               // которые можно отменить; при переходе к
                               // другой схеме буфер отмены сбрасывается
#define  KOL_ELEM         900  // 0 - 897: обычные элементы
                               // 898, 899: ВЫХОД и КОНЕЦ
#define  MAX_KOL_ELEM     KOL_ELEM-2  // максимальное количество элементов

#define  KOL_STR_KOMMENT  10
#define  KOL_STR_DEJSTW   20

#define  KOL_WETOK        50   // максимальное количество веток силуэта
#define  KOL_WERT         KOL_WETOK + KOL_ELEM      // максимальное количество
                                                    // вертикалей силуэта
#define  MAX_KOL_WERT     KOL_WERT-1  // при проверке на добавление новой
                                      // вертикали
                               // у последнего элемента массива вертикалей
                               // всегда сброшен  used
                               // последняя вертикаль - для ветки ВЫХОД
#define  T_WERT_GLAWNAQ   1
#define  T_WERT_DOCHX     2
#define  T_WERT_CIKLA     3

#define  KOL_OTKR_WERT    100  // число открытых и одновременно
                               // создаваемых вертикалей
#define  MAX_OTKR_WERT    KOL_OTKR_WERT-1

#define  OTSTUP_WERT      30   // расстояние по X между вертикалями
#define  DALVJAZ_Y_ZAGOL   7   // расстояние от заголовка от начала координат
#define  DALVJAZ_Y_ZAGOL2  10

#define  SM_X_PIKTOGRAMMY  11  // смещение пиктограммы в пикселях от левой
                               // границы блока
#define  SM_Y_PIKTOGRAMMY  2   // смещение пиктограммы в пикселях от верхней
                               // границы блока
#define  RAZM_M_BITMAP    10   // размер массива рисунков

#define  RAZM_X_BMP       21   // размеры пиктограммы по X и У
#define  RAZM_Y_BMP       20


#define  GRANICA_N_EL     58   // при создании shema.prg
#define  L_GR_TEKSTA      5    // левая граница текста при создании shema.prg


#define  RAZM_LSW         KOL_ELEM    // макс. число элементов ЛСВ
#define  RAZM_BUF_LSW     100         // размер буфера копирования эл. ЛСВ
#define  RAZM_STOTM_LSW   10          // размер стека отмены ввода ЛСВ
#define  RAZM_STEKA_LSW   100         // размер стека ЛСВ, используемого
                                      // при генерации схемы

// ошибки создания схемы

#define  SHEMA_OK               0  // нет ошибок

                                   // для первых 12 ошибок текущим эл-том
                                   // является 2-й, для которого for

                                   // если в стеке ЛСВ есть элементы,
                                   // но верхний эл. стека неправильный
#define  SHEMA_IF_for_CE        1  // "Незакрытый IF для CE"
#define  SHEMA_EI_for_CE        2  // "Незакрытый EI для CE"
#define  SHEMA_EB_for_CE        3  // "Незакрытый EB для CE"
#define  SHEMA_CB_for_EI        4  // "Незакрытый CB для EI"
#define  SHEMA_CB_for_EB        5  // "Незакрытый CB для EB"
```

```
#define  SHEMA_CB_for_E          6  // "Незакрытый СВ для Е"
#define  SHEMA_powtor_EB         7  // "Повтор ЕВ"
#define  SHEMA_EB_for_EI         8  // "Незакрытый ЕВ для ЕI"

                                    // если в стеке ЛСВ нет элементов,
                                    // а нужно проверить верхний элемент стека
#define  SHEMA_no_IF_for_EI      9  // "Нет IF для ЕI"
#define  SHEMA_no_IF_for_EB      10  // "Нет IF для ЕВ"
#define  SHEMA_no_IF_for_E       11 // "Нет IF для Е"
#define  SHEMA_no_CB_for_CE      12 // "Нет СВ для СЕ"
                                    //=====================================
#define  SHEMA_OSH_STEK_FULL     13 // "Начало блока: стек полон"
                                    // стек ЛСВ заполнен при попытке положить
                                    // туда очередной элемент
#define  SHEMA_OSH_STEK_EMPTY    14 // "Конец блока: стек пуст"
                                    // стек ЛСВ пуст при попытке взять оттуда
                                    // верхний элемент
#define  SHEMA_net_el_wetki      15 // "схема: нет эл. ветки "
                                    // не найдены элементы для текущей ветки
                                    // при удалении ссылок на вертикали для
                                    // элементов текущей ветки
#define  SHEMA_poterq_el         16 // "схема: потеря элементов"
                                    // при удалении ссылок на вертикали для
                                    // элементов текущей ветки обнаружено,
                                    // что эл. схемы меньше, чем должно быть
#define  SHEMA_net_w_wetki       17 // "схема: нет верт. ветки "
                                    // не найдены дополнительные вертикали
                                    // ветки, хотя они должны быть
#define  SHEMA_poterq_dwert      18 // "схема: потеря вертикалей"
                                    // потеря дополнительных вертикалей у ветки
                                    // меньше, чем должно быть
#define  SHEMA_el_w_steke_LSW    19 // "остались эл. в стеке ЛСВ"
                                    // остались эл. в стеке после окончания
                                    // сканирования ЛСВ
#define  SHEMA_m_elem_full       20 // "массив элементов полон"
                                    // при попытке добавить вмассив элементов
                                    // новый элемент оказалось, что он уже
                                    // заполнен
#define  SHEMA_m_wert_full       21 // "массив вертикалей полон"
                                    // при попытке добавить в массив вертикалей
                                    // новую вертикаль оказалось, что
                                    // массив вертикалей уже заполнен
#define  SHEMA_net_gran_el_w     22 // "нет граничных эл.вертикали"
                                    // не найден родительский или конечный эл.
                                    // для вертикали после сортировки эл. схемы
                                    // в конце создания схемы
#define  SHEMA_m_wetok_full      23 // "массив веток полон"
                                    // при попытке вставить новую ветку
                                    // оказалось, что массив веток заполнен
#define  SHEMA_m_otkr_wert_full  24 // "мас.откр.верт. полон"
                                    // при попытке добавить в массив открытых
                                    // вертикалей новую вертикаль оказалось,
                                    // что мас.откр.верт. уже заполнен
#define  SHEMA_no_otkr_wert      25 // "не найдена откр.вертикаль"
                                    // при попытке перейти к вертикали,
                                    // созданной IF или EI эта вертикаль не
                                    // найдена в массиве открытых верт.
#define  SHEMA_osh_ind_mow       26 // "ошиб.инд.мас.откр.верт."
                                    // ошибка индекса мас. открытых верт.
                                    // (i < 1) или (i > MAX_OTKR_WERT) при
                                    // попытке добавить эл. в мас.откр. верт.
#define  SHEMA_dmow_full         27 // "доп.мас.откр.верт. полон"
                                    // когда t_E помещал вертикали блока t_IF
                                    // в доп.мас.откр.верт., оказалось, что
```

```
                                    // этот массив уже заполнен

// ошибки чтения файла

#define  FAJL_net_zagol_shemy    28 // "файл: нет загол. схемы"
                                    // при чтении shema.prg прочитана
                                    // непустая строка - не заголовок схемы
#define  FAJL_net_zagol_w1       29 // "файл: нет загол. ветки 1"
                                    // при чтении shema.prg
                                    // первый элемент ветки 1 - не заголовок



typedef  struct {
    char    m[dl_stro];
}  str210;


// тип для вертикали

typedef  struct {

int    wetka;       // номер ветки, к которой относится вертикаль
uchar  used;        // IS/NO
int    type;        // главная, дочерняя, вертикаль цикла
int    parent_w;    // индекс родительской вертикали или -1
int    parent_el;   // индекс родительского элемента или -1
int    wet_pel;     // ветка и номер строки ЛСВ родительского элемента, нужны
int    n_str_pel;   //  для нахождения нового индекса родительского эл-та
                    //  после сортировки элементов ветки по n_str
                    //  в конце создания ветки
int    parent_endw; // индекс родительской вертикали для элемента конца
int    parent_end;  // индекс конца, закрывающего дочернюю вертикаль или -1
int    wet_pend;    // аналогично wet_pel и n_str_pel
int    n_str_pend;  //
int    x;           // координаты вертикали в системе координат схемы
int    y1/*,y2*/;
int    tyel;        // координата У следующего эл. в системе коорд. вертикали
                    // в процессе создания вертикали
                    // после создания вертикали - нижняя точка вертикали в
                    // системе координат вертикали
int    n_sprawa;    // номер справа от главн. верт. или 0
int    type_1_el;   // тип 1-го элемента для дочерней верт.
                    // если t_EB, то линия до 1-го блока рисуется сама,
                    // иначе нужно рисовать вручную
}  t_wert;


// тип для элемента силуэта

typedef  struct  {

    char  name[rname_el]; // имя элемента
    int   ind;            // индекс элемента на схеме и в списке ЛСВ
    int   wetka;          // номер ветки
    int    wetka2;        // номер ветки перехода для t_AD,
                          // если не задана то == -1;   0 - на выход
    uchar fl_usl;         // IS/NO, флаг условия для IF, EI, CE
    int   n_str;          // индекс по порядку следования в ЛСВ
    uchar used;           // IS/NO
    uchar old;            // перед обновлением ветки все ее эл. помечаются
                          // как старые, а те, которые остаются старыми
                          // после окончания обновления ветки, удаляются
```

```
    int   type;           // тип элемента
    int   n_wert;         // номер вертикали
    int   n_na_wert;      // номер элемента на вертикали
    int   n_dwert;        // номер дочерней вертикали
    int   x1,x2,y1,y2;    // координаты в системе координат ветки
    uchar end_w;          // IS/NO - эл. является одним из окончаний ветки

    str210 m_d[KOL_STR_DEJSTW];  // массив строк действий
    int kol_s_d;                 // число строк действий
    str210 m_k[KOL_STR_KOMMENT]; // массив строк комментариев
    int kol_s_k;                 // число строк комментариев
    int  n_bmp;           // -1: bmp-файл к элементу не привязан

}  t_elem;


typedef  struct  {
char     imq_zagol[dl_stro];

t_elem   m_el[KOL_ELEM];    // массив элементов силуэта

//   Перед созданием ветки для текущих элементов ветки удаляются ссылки
// на вертикали ветки, а затем удаляются все вертикали ветки.
//   После окончания создания ветки будут удалены те элементы ветки, которые
// остались не привязанными к вертикалям.
//
//   Т.к. элементы удаляются и вставляются хаотически (путем сброса и
// взведения флага used), то нельзя поручиться, что и элементы одной метки
// будут следовать друг за другом в массиве элементов по порядку их
// следования в ЛСВ.
//   Поэтому для заново построенной ветки:
// 1) находим индексы элементов ветки в массиве элементов;
// 2) сортируем элементы ветки по порядку их следования в ЛСВ.


int      kol_elem;          // число элементов силуэта

int kol_wetok;              // число веток
                            // последняя ветка - ветка выхода,
                            // состоит их элементов ВЫХОД и КОНЕЦ,
                            // размещаемых в конце массива m_el
int m_el_wetki[KOL_WETOK];      // элементов на ветке
int m_old_el_wetki[KOL_WETOK];  // старое число элементов
int m_dw_wetki[KOL_WETOK];  // число дополнительных вертикалей ветки
int m_inewet[KOL_WETOK];    // массив индексов начальных
                            // элементов веток
int m_inwwet[KOL_WETOK];    // массив индексов начальных дополнительных
                            // вертикалей веток
                            // если индекс массива = -1, то у ветки нет
                            // дополнительных вертикалей
int m_x_wetok[KOL_WETOK];   // координаты X веток
int m_dx_wetok[KOL_WETOK];  // размеры по X для веток
                            // слева от главной вертикали i-я ветка занимает
                            // (6 символов)*(ширина символа)
                            // справа от главной вертикали i-я ветка занимает
                            // m_dx_wetok[i]-((6 символов)*(ширина символа))
                            // расстояние между соседними ветками
                            // равно OTSTUP_WERT, столько же, сколько и
                            // отступ 1-й ветки от начала координат
                            // если у ветки только главная вертикаль, то
                            // справа от главной вертикали i-я ветка занимает
                            // (3 символов)*(ширина символа)

t_wert  m_wert[KOL_WERT];   // массив вертикалей силуэта
```

```
                            // первые KOL_WETOK вертикалей отводятся для
                            // хранения главных вертикалей веток
                            // дополнительные вертикали веток хранятся
                            // начиная со смещения KOL_WETOK
int kol_dwert;              // число вертикалей
                            // число дополнительных вертикалей равно
                            // kol_wert - KOL_WETOK
int m_el_wert[KOL_WERT];    // элементов на вертикали

int m_inew[KOL_WERT];       // массив индексов начальных
                            // элементов вертикалей




}  t_alg;



// ЛСВ - логическая структура ветки

typedef  struct {

uchar  used;      // IS/NO
int    n_str;     // номер строки в списке ЛСВ
int    ind;       // порядковый номер на схеме; 0 - если новый элемент
int    type;
int    ind_m_el;  // индекс в массиве m_el[KOL_ELEM] или -1 для нового эл.

}  el_lsw;



typedef  struct {

el_lsw   m_lsw[RAZM_LSW];
int      kol_m_lsw;

} el_stotm;  // элемент стека отмены ввода ЛСВ



extern  t_alg   tek_alg;       // текущий алгоритм

//extern  t_alg   m_otmena[RAZM_OTMENA];
//extern  int     kol_otmen;                 // число позиций в буфере отмены

extern  el_lsw  m_lsw[RAZM_LSW];           // массив элементов ЛСВ
extern  int     kol_m_lsw;
extern  int     ind_top_lsw;               // индекс верхней видимой строки ЛСВ
extern  int     ind_sel_lsw;               // индекс выделенной строки ЛСВ

extern  el_stotm  stotm[RAZM_STOTM_LSW];   // стек отмены ввода ЛСВ
extern  int       kol_stotm;
extern  el_lsw  buf_lsw[RAZM_BUF_LSW];     // буфер копирования ЛСВ
extern  int     kol_b_lsw;

extern  el_lsw  stek_lsw[RAZM_STEKA_LSW];  // стек ЛСВ
extern  int     kol_s_lsw;
```

```
extern  el_lsw  tek_el_shemy;     // текущий элемент при построении схемы
extern  el_lsw  el_steka_lsw;     // для чтения эл-та из стека ЛСВ

extern  int  m_otkr_wert[KOL_OTKR_WERT];  // массив открытых и одновременно
                                          // создаваемых вертикалей
                              // этот массив содержит индексы открытых
                              // вертикалей из массива tek_alg.m_wert
                              // при создании схемы какое-либо конкретное
                              // значение n_sprawa может принадлежать только
                              // одной создаваемой вертикали, хотя после
                              // закрытия этой вертикали другая создаваемая
                              // вертикаль может снова получить такое же
                              // значение n_sprawa
extern  int     kol_otkr_wert; // число одновременно открытых вертикалей
extern  int     tek_otkr_wert; // текущая открытая вертикаль

extern  int mie_wetki[KOL_ELEM];  // массив индексов элементов ветки
                            // нужен для сортировки элементов ветки в порядке
                            // их следования в ЛСВ

//-----------------------------------------------------------------------

extern  int  dalvjaz_chw;            // ширина символа текущего фонта
extern  int  dalvjaz_chh;            // высота символа текущего фонта
extern  int  dalvjaz_chdy;           // шаг по y между двумя строками

extern  int  dalvjaz_smx;            // смещение БС по x
extern  int  dalvjaz_smy;            // смещение БС по y
extern  int  dalvjaz_oknox;          // размер окна работы с БС по x
extern  int  dalvjaz_oknoy;          // размер окна работы с БС по y
extern  int  dalvjaz_xmax;           // максимальная x для схемы
extern  int  dalvjaz_ymax;           // максимальная y для схемы
extern  int  dalvjaz_tek_wetka;      // текущая ветка для окна силуэта

extern  uchar  dalvjaz_shema_schitana; // схема была считана из файла целиком,
                                       // на структурность проверены все ветки


extern  int  dalvjaz_fl_bmp;         // при создании bmp фон белый

extern  char  cfg_imq_fonta_1[dl_stro];        //_имя_фонта_1
extern  char  cfg_razmer_fonta_1[dl_stro];     //_размер_фонта_1

extern  char m_imen_bmp[RAZM_M_BITMAP][dl_stro];  // имена файлов пиктограмм

extern  int  dalvjaz_beg;

extern  int  fl_otkata_shemy;

//-------------------------------------------------------------------------

void  el_type_to_str(int type, char *stro);
uchar  str_probelow(char *str);

//-------------------------------------------------------------------------
#endif
```

ПРИЛОЖЕНИЕ В. U_LS.CPP – обработка ЛСВ в dal_vjaz

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include <process.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mem.h>
#include <dir.h>
#include "u_dalvj.h"
#include "u_ls.h"
#include "utils.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
Tform_ls *form_ls;


t_alg      tek_alg;       // текущий алгоритм

//t_alg      m_otmena[RAZM_OTMENA];
//int        kol_otmen;                  // число позиций в буфере отмены

el_lsw     m_lsw[RAZM_LSW];           // массив элементов ЛСВ
int        kol_m_lsw;
int        ind_top_lsw;               // индекс верхней видимой строки ЛСВ
int        ind_sel_lsw;               // индекс выделенной строки ЛСВ

el_lsw     m_lsw2[RAZM_LSW];          // для восстановления после отката
int        kol_m_lsw2;                // см. wernutx_sostoqnie


el_stotm   stotm[RAZM_STOTM_LSW];     // стек отмены ввода ЛСВ
int        kol_stotm;
el_lsw     buf_lsw[RAZM_BUF_LSW];     // буфер копирования ЛСВ
int        kol_b_lsw;

el_lsw     stek_lsw[RAZM_STEKA_LSW];  // стек ЛСВ
int        kol_s_lsw;

el_lsw     tek_el_shemy;     // текущий элемент при построении схемы
el_lsw     el_steka_lsw;     // для чтения эл-та из стека ЛСВ

int        m_otkr_wert[KOL_OTKR_WERT];  // массив открытых и одновременно
                                        // создаваемых вертикалей
                              // этот массив содержит индексы открытых
                              // вертикалей из массива tek_alg.m_wert
                              // при создании схемы какое-либо конкретное
                              // значение n_sprawa может принадлежать только
                              // одной создаваемой вертикали, хотя после
                              // закрытия этой вертикали другая создаваемая
                              // вертикаль может снова получить такое же
                              // значение n_sprawa
int        kol_otkr_wert;     // число одновременно открытых вертикалей
int        tek_otkr_wert;     // текущая открытая вертикаль

int mie_wetki[KOL_ELEM];      // массив индексов элементов ветки
```

```
                                // нужен для сортировки элементов ветки в порядке
                                // их следования в ЛСВ

int  dalvjaz_chw;               // ширина символа текущего фонта
int  dalvjaz_chh;               // высота символа текущего фонта
int  dalvjaz_chdy;              // шаг по у между двумя строками

int  dalvjaz_smx;               // смещение БС по х
int  dalvjaz_smy;               // смещение БС по у
int  dalvjaz_oknox;             // размер окна работы с БС по х
int  dalvjaz_oknoy;             // размер окна работы с БС по у
int  dalvjaz_xmax;              // максимальная х для схемы
int  dalvjaz_ymax;              // максимальная у для схемы
int  dalvjaz_tek_wetka = 0;     // текущая ветка для окна силуэта

uchar  dalvjaz_shema_schitana = NO; // схема была считана из файла целиком,
                                    // на структурность проверены все ветки

int  dalvjaz_fl_bmp = 0;        // при создании bmp фон белый


char  cfg_imq_fonta_1[dl_stro];         //_имя_фонта_1
char  cfg_razmer_fonta_1[dl_stro];      //_размер_фонта_1

char m_imen_bmp[RAZM_M_BITMAP][dl_stro];  // имена файлов пиктограмм

int  dalvjaz_beg = 0;

int  fl_otkata_shemy = NO;

//---------------------------------------------------------------------------

void  push_stek_lsw();
void  pop_stek_lsw();
void  give_stek_lsw(int n);
void  sbros_steka_lsw();
void  mow__init(int tw);
int   mow__n_wert(int n);
void  mow__wstawitx(int tw, int n);
void  mow__udalitx(int n);

//---------------------------------------------------------------------------

// i.2. конструктор формы ЛСВ

__fastcall Tform_ls::Tform_ls(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------

// i.3. действия при создании формы ЛСВ

void __fastcall Tform_ls::FormCreate(TObject *Sender)
{
          int n, i;

          Height = 710;

          l_soob_oshibka->Visible = false;
          l_soob_oshibka->Left = 8;
          l_soob_oshibka->Top = 576;
          l_soob_no_osh->Visible = false;
          l_soob_no_osh->Left = 8;
```

```
            l_soob_no_osh->Top = 576;

            Show();

            l_soob->Visible = false;
}
//---------------------------------------------------------------------------

// i.4. самостоятельное рисование строки списка ЛСВ как строки таблицы

void __fastcall Tform_ls::ListBox1DrawItem(TWinControl *Control, int Index,
      TRect &Rect, TOwnerDrawState State)
{
  Graphics::TBitmap *pBitmap; // temporary variable for item's bitmap
  int     Offset = 2;   // default text offset width
  int i, sm, lb_str, chw, j, fl_ind, bm, em, smy, l2, dl2, fl_pl;
  char stro[210/*dl_stro*/];
                                                                    //.1
  // note that we draw on the listbox's canvas, not on the form
  TCanvas *pCanvas = ((TListBox *)Control)->Canvas;
  pCanvas->FillRect(Rect); // clear the rectangle
  pBitmap = (Graphics::TBitmap *)((TListBox *)Control)->Items->Objects[Index];

  if (pBitmap)
  {
    pCanvas->BrushCopy(Bounds(Rect.Left + Offset, Rect.Top, pBitmap->Width,
pBitmap->Height), pBitmap, Bounds(0, 0, pBitmap->Width, pBitmap->Height),
clRed); // render bitmap
    Offset += pBitmap->Width + 4;   // add four pixels between bitmap and text}
  }

  strcpy(stro, ListBox1->Items->Strings[Index].c_str());

  for (j=0; j < strlen(stro); j++)
        if (stro[j] == '|')   stro[j] = ' ';

  pCanvas->TextOut(Rect.Left + Offset, Rect.Top,
      AnsiString(stro/*mal[MDItek].m_alg[i]->msalg[sm].m*/));

  chw = pCanvas->TextWidth("0");
  smy = Rect.Bottom-Rect.Top + 1;

  strcpy(stro, ListBox1->Items->Strings[Index].c_str());

  for (j=0; j < strlen(stro); j++) {
    if (stro[j] == '|')  {

      pCanvas->MoveTo(Rect.Left+Offset+(chw*j)+(chw/2), Rect.Bottom+1-smy);
      pCanvas->LineTo(Rect.Left+Offset+(chw*j)+(chw/2), Rect.Bottom+1);
    }
  }

}
//---------------------------------------------------------------------------

// i.5. обработка клавиатуры для списка ЛСВ
//      клавиши:  F2 - F8, F10, Delete, Insert, Ctrl+X, Ctrl+C, Ctrl+V
//      "+" цифровой группы, "-" цифровой группы

void __fastcall Tform_ls::ListBox1KeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
         int i, ii, kol, ti, w_buf, si,
             fl_Ctrl_C = 0;  // 1 - только копирование в буфер
```

```
WORD Key2;


if (l_soob_oshibka->Visible == true)
                        l_soob_oshibka->Visible = false;

Key2 = Key;

if (Key2 == k_F10) {

     b_sozdatx_shemuClick(Sender);   return;  }


if ((Key2 == 67) &&              // C
   (Shift.Contains(ssCtrl))) { Key2 = k_Del;  fl_Ctrl_C = 1;  }
if ((Key2 == 88) &&              // X
   (Shift.Contains(ssCtrl)))   Key2 = k_Del;
if ((Key2 == 86) &&              // V
   (Shift.Contains(ssCtrl)))   Key2 = k_Ins;


// переход к другим веткам по "+" и "-" группы цифровых клавиш

if (Key2 == k_cifr_PLUS)
                form_dalvjaz->b_w_plClick(Sender);

if (Key2 == k_cifr_MIN)
                form_dalvjaz->b_w_minClick(Sender);

// изменение ЛСВ

if (((Key2 >= k_F2) && (Key2 <= k_F9)) ||
    (Key2 == k_Del) || (Key2 == k_Ins)) {

  kol = 0;
  for (i=0; i < ListBox1->Items->Count; i++) {

    if (ListBox1->Selected[i]) {

      if (i == 0)  return;  // действия с 1-й строкой списка запрещены

      if (kol == 0) {

        kol = 1;  ii = i;

        lsw_selected = ii;  // 1-я выделенная строка
        lsw_kol_sel  = 1;
      }
      else  {
        if (i != (ii+1))  return;  // строки выделены не подряд

        kol++;   ii = i;  }  // последняя выделенная строка
    }

    lsw_kol_sel = kol;
  }

  if ((kol == 1) || (Key2 == k_Del))  {////

  if ((Key2 == k_Del) && (ii >= kol_m_lsw))   return;


  // сохраняем список в стеке отмены
```

```
for (i= (RAZM_STOTM_LSW-1); i > 0; i--) {

  stotm[i].kol_m_lsw = stotm[i-1].kol_m_lsw;
  memcpy(stotm[i].m_lsw, stotm[i-1].m_lsw, sizeof(el_lsw)*RAZM_LSW);
}

stotm[0].kol_m_lsw = kol_m_lsw;
memcpy(stotm[0].m_lsw, m_lsw, sizeof(el_lsw)*RAZM_LSW);

if (kol_stotm < 10)  kol_stotm++;


if ((Key2 != k_Del) && (Key2 != k_Ins)) {

  if (kol_m_lsw < RAZM_LSW) {

    for (i=kol_m_lsw; i > ii; i--)
                memcpy(&m_lsw[i], &m_lsw[i-1], sizeof(el_lsw));
    kol_m_lsw++;

    m_lsw[i].ind      = 0;
    m_lsw[i].ind_m_el = -1;

    if (Key2 == k_F2)   m_lsw[i].type = t_IF;
    if (Key2 == k_F3)   m_lsw[i].type = t_EI;
    if (Key2 == k_F4)   m_lsw[i].type = t_EB;
    if (Key2 == k_F5)   m_lsw[i].type = t_E;
    if (Key2 == k_F6)   m_lsw[i].type = t_CB;
    if (Key2 == k_F7)   m_lsw[i].type = t_CE;
    if (Key2 == k_F8)   m_lsw[i].type = t_A;
    if (Key2 == k_F9)   m_lsw[i].type = t_AD;
  }
}
else {// Del

  if (Key2 == k_Del) {

    w_buf = lsw_kol_sel;
    if (w_buf > RAZM_BUF_LSW)  w_buf = RAZM_BUF_LSW;

    for (i=0; i < w_buf; i++) {

      memcpy(&buf_lsw[i], &m_lsw[lsw_selected+i], sizeof(el_lsw));

      buf_lsw[i].ind = 0;
    }
    kol_b_lsw = w_buf;

    if (fl_Ctrl_C == 1) {

      for (i=0; i < w_buf; i++) {

        // в буфере будут новые элементы
        buf_lsw[i].ind = 0;   buf_lsw[i].ind_m_el = -1;
      }

      fl_Ctrl_C = 0;
    }
    else {
      // удаление из ЛСВ

      for (i = lsw_selected; i < (kol_m_lsw - lsw_kol_sel); i++)
          memcpy(&m_lsw[i], &m_lsw[i+lsw_kol_sel], sizeof(el_lsw));
```

```
        kol_m_lsw -= lsw_kol_sel;
      }
    }
    else {// k_Ins

      if (kol_b_lsw == 0)  return;

      if ((kol_b_lsw + kol_m_lsw) > RAZM_LSW)   return;

      for (i=(kol_m_lsw-1); i >= lsw_selected; i--)
          memcpy(&m_lsw[i+kol_b_lsw], &m_lsw[i], sizeof(el_lsw));

      for (i=lsw_selected; i < (lsw_selected+kol_b_lsw); i++)
          memcpy(&m_lsw[i], &buf_lsw[i-lsw_selected], sizeof(el_lsw));

      kol_m_lsw += kol_b_lsw;


      for (i=0; i < kol_b_lsw; i++) {

        // в буфере будут новые элементы
        buf_lsw[i].ind = 0;   buf_lsw[i].ind_m_el = -1;
      }

      //kol_b_lsw = 0;
    }
  }

  ti = ListBox1->TopIndex;

  chtenie_LSW(Sender);

  ListBox1->TopIndex = ti;

  si = 0;
  if ((Key2 >= k_F2) && (Key2 <= k_F9))   si = 1;

  ListBox1->Selected[lsw_selected+si] = true;

  }////  if ((kol == 1) || (Key2 == k_Del))
}

if (Key2 == k_Backspace) {

  if (kol_stotm == 0)  return;

  kol_m_lsw = stotm[0].kol_m_lsw;
  memcpy(m_lsw, stotm[0].m_lsw, sizeof(el_lsw)*RAZM_LSW);

  for (i=0; i < (kol_stotm-1); i++) {

    stotm[i].kol_m_lsw = stotm[i+1].kol_m_lsw;
    memcpy(stotm[i].m_lsw, stotm[i+1].m_lsw, sizeof(el_lsw)*RAZM_LSW);
  }

  kol_stotm--;


  ti = ListBox1->TopIndex;

  chtenie_LSW(Sender);

  ListBox1->TopIndex = ti;
```

```
          ListBox1->Selected[lsw_selected] = true;
        }
}
//---------------------------------------------------------------------------

// i.6. начальная инициализация после создания формы ВС
//      синхронизация положения с окном ВС

void __fastcall Tform_ls::Timer1Timer(TObject *Sender)
{
  if (dalvjaz_beg == 1) {

    dalvjaz_beg = 0;

    //---------- инициализация основных переменных ---------------

    strcpy(cfg_imq_fonta_1, "Courier New");
    strcpy(cfg_razmer_fonta_1, "9");

    //kol_otmen = 0;  // сброс массива отмен
    kol_b_lsw = 0;  // сброс буфера переноса ЛСВ


    //-------------------------------------------------------------


    //form_dalvjaz->init_alg(Sender);

    form_ls->l_soob->Visible = true;

    form_ls->Timer1->Enabled = true;

    form_ls->zagruzka_LSW(Sender);

    form_dalvjaz->TabControl1->Tabs->Clear();

    form_dalvjaz->TabControl1->Tabs->Add("shema.prg");

    form_dalvjaz->b_chtenieClick(Sender);

    form_dalvjaz->Show();

    form_ls->SetFocus();


    //form_dalvjaz->wywod_shemy(Sender);

  }// if (dalvjaz_beg == 1)



  if ((Left != form_dalvjaz->Left - Width) ||
      (Top != form_dalvjaz->Top)) {
    Top = form_dalvjaz->Top;
    Left = form_dalvjaz->Left - Width;
  }
}
//---------------------------------------------------------------------------

// i.7.


// i.8.
```

```
// i.9. создать из списка ЛСВ ветку визуальной схемы

void __fastcall Tform_ls::b_sozdatx_shemuClick(TObject *Sender)
{
         int i, tw;


         for (i=0; i < ListBox1->Items->Count; i++) {

            if (ListBox1->Selected[i])  {  lsw_selected = i;   break;   }
         }



         //tek_alg.kol_dwert = tek_alg.kol_wetok-1;

         tek_alg.m_wert[dalvjaz_tek_wetka].type = T_WERT_GLAWNAQ;
         tek_alg.m_wert[dalvjaz_tek_wetka].used = IS;
         tek_alg.m_wert[dalvjaz_tek_wetka].x  =
                                    tek_alg.m_x_wetok[dalvjaz_tek_wetka];

         tek_alg.m_inew[dalvjaz_tek_wetka] = -1;

         tek_alg.m_el_wert[dalvjaz_tek_wetka] = 0;

         tek_alg.m_wert[dalvjaz_tek_wetka].tyel =
                                              dalvjaz_chdy + (dalvjaz_chdy/2);

         tek_alg.m_wert[dalvjaz_tek_wetka].y1 =
                        DALVJAZ_Y_ZAGOL + ((5*dalvjaz_chdy)/2);

         mow__init(dalvjaz_tek_wetka);

         // главная вертикаль ветки будет открыта вплоть до
         // завершения сканирования ЛСВ

         //-------------------------------------------

         tw = dalvjaz_tek_wetka;

         zapomnitx_sostoqnie(Sender);

         tek_str_lsw = 0;

         osh_sozd_shemy = SHEMA_OK;

         udal_wert_tek_wetki(Sender);

         if (osh_sozd_shemy != SHEMA_OK)  goto  m_osh;

         // сканируем ЛСВ


         sbros_steka_lsw();


         for (i=0; i < kol_m_lsw; i++) {

           tek_str_lsw = i;

           memcpy(&tek_el_shemy, &m_lsw[i], sizeof(el_lsw));
           tek_el_shemy.n_str = i;
```

```
  give_stek_lsw(0);

  prow_oshibki_struktury(Sender);
  if (osh_sozd_shemy != SHEMA_OK)  goto  m_osh;

  if ((tek_el_shemy.type == t_IF) ||
      (tek_el_shemy.type == t_EI) ||
      (tek_el_shemy.type == t_CB)) {

    if (tek_el_shemy.type == t_IF)   obr_shemy_dlq_IF(Sender);
    if (tek_el_shemy.type == t_EI)   obr_shemy_dlq_EI(Sender);
    if (tek_el_shemy.type == t_CB)   obr_shemy_dlq_CB(Sender);
    if (osh_sozd_shemy != SHEMA_OK)   goto  m_osh;

    push_stek_lsw();
  }
  else
  if (tek_el_shemy.type == t_EB) {
    obr_shemy_dlq_EB(Sender);
    if (osh_sozd_shemy != SHEMA_OK)   goto  m_osh;
  }
  else
  if (tek_el_shemy.type == t_E) {

    // извлекаем из стека, пока не извлечем IF или
    // пока стек не станет пустым

    obr_shemy_dlq_E(Sender);
    if (osh_sozd_shemy != SHEMA_OK)   goto  m_osh;

    while (1) {

      if (el_steka_lsw.used == NO) {

        osh_sozd_shemy = SHEMA_no_IF_for_E;   goto  m_osh;  }

      pop_stek_lsw();

      if (el_steka_lsw.type == t_IF)
                                     break;
    }
    //

  }
  else
  if (tek_el_shemy.type == t_CE) {

    // извлекаем из стека, пока не извлечем IF или
    // пока стек не станет пустым

    obr_shemy_dlq_CE(Sender);
    if (osh_sozd_shemy != SHEMA_OK)   goto  m_osh;

    if (el_steka_lsw.used == NO) {

      osh_sozd_shemy = SHEMA_no_CB_for_CE;   goto  m_osh;  }

    pop_stek_lsw();
  }
  else {
    obr_shemu_dlq_HD_A_AD(Sender);
    if (osh_sozd_shemy != SHEMA_OK)   goto  m_osh;
  }
}// конец сканирования ЛСВ
```

```
          give_stek_lsw(0);

          if (el_steka_lsw.used == IS) {
            osh_sozd_shemy = SHEMA_el_w_steke_LSW;   goto  m_osh;  }

          obr_shemy_w_konce_lsw(Sender);

          if (osh_sozd_shemy == SHEMA_OK) {

            if (dalvjaz_shema_schitana == IS) {

              // нужен пересчет координат для всех веток

              pereschet_koord_el(Sender);

              form_dalvjaz->wywod_shemy(Sender);
              form_ls->SetFocus();
              zagruzka_LSW(Sender);
              ListBox1->Selected[lsw_selected] = true;
            }
          }

          m_osh:    obrabotka_oshibok(Sender);
}
//---------------------------------------------------------------------------

// i.10. удалить вертикали текущей ветки

void __fastcall Tform_ls::udal_wert_tek_wetki(TObject *Sender)
{

          int i, j, tw, kdw, tek, kol;

          tw = dalvjaz_tek_wetka;


          // удаление ссылок на вертикали для элементов текущей ветки
          // помечаем элементы ветки как старые

          i = 0;  tek = 0;  kol = 0;

          while ((tek < tek_alg.kol_elem) && (i < MAX_KOL_ELEM)) {

            if (tek_alg.m_el[i].used == IS) {

              tek++;

              if (tek_alg.m_el[i].wetka == (tw+1))  {

                tek_alg.m_el[i].old = IS;

                tek_alg.m_el[i].n_wert = -1;   tek_alg.m_el[i].n_dwert = -1;

                kol++;  }
            }

            i++;
          }

          if (kol == 0)  {

            osh_sozd_shemy = SHEMA_net_el_wetki;
            goto  m_osh;
```

```
          }
          if (tek < tek_alg.kol_elem) {

            osh_sozd_shemy = SHEMA_poterq_el;
            goto  m_osh;
          }

          tek_alg.m_old_el_wetki[tw] = tek_alg.m_el_wetki[tw];

          tek_alg.m_el_wetki[tw]     = 0;   // число элементов для ветки
                                            // формируем заново

          // удаление дополнительных вертикалей текущей ветки

          if (tek_alg.kol_dwert > 0) {

            i = KOL_WETOK;  tek = 0;  kol = 0;

            while ((tek < tek_alg.kol_dwert) && (i < MAX_KOL_WERT)) {

              if (tek_alg.m_wert[i].used == IS) {

                tek++;

                if (tek_alg.m_wert[i].wetka == (tw+1))  {

                  tek_alg.m_wert[i].used = NO;  kol++;  }
              }

              i++;
            }

            if (tek < tek_alg.kol_dwert) {

              osh_sozd_shemy = SHEMA_poterq_dwert;
              goto  m_osh;
            }

            tek_alg.kol_dwert = tek_alg.kol_dwert - kol;
            tek_alg.m_dw_wetki[tw] = 0;
          }

          m_osh:

}



// i.11. обработка ошибок

void __fastcall Tform_ls::obrabotka_oshibok(TObject *Sender)
{
          int i;

          if (osh_sozd_shemy != SHEMA_OK) {

            if (osh_sozd_shemy == SHEMA_no_IF_for_EI)    //
               l_soob_oshibka->Caption = "Нет IF для EI";

            if (osh_sozd_shemy == SHEMA_no_IF_for_EB)    //
               l_soob_oshibka->Caption = "Нет IF для EB";

            if (osh_sozd_shemy == SHEMA_no_IF_for_E)     //
```

```
   l_soob_oshibka->Caption = "Нет IF для E";

if (osh_sozd_shemy == SHEMA_no_CB_for_CE)    //
   l_soob_oshibka->Caption = "Нет CB для CE";

if (osh_sozd_shemy == SHEMA_IF_for_CE)       //
   l_soob_oshibka->Caption = "Незакрытый IF для CE";

if (osh_sozd_shemy == SHEMA_EI_for_CE)       //
   l_soob_oshibka->Caption = "Незакрытый EI для CE";

if (osh_sozd_shemy == SHEMA_EB_for_CE)       //
   l_soob_oshibka->Caption = "Незакрытый EB для CE";

if (osh_sozd_shemy == SHEMA_CB_for_EI)       //
   l_soob_oshibka->Caption = "Незакрытый CB для EI";

if (osh_sozd_shemy == SHEMA_CB_for_EB)       //
   l_soob_oshibka->Caption = "Незакрытый CB для EB";

if (osh_sozd_shemy == SHEMA_CB_for_E)        //
   l_soob_oshibka->Caption = "Незакрытый CB для E";

if (osh_sozd_shemy == SHEMA_powtor_EB)       //
   l_soob_oshibka->Caption = "Повтор EB";

if (osh_sozd_shemy == SHEMA_EB_for_EI)       //
   l_soob_oshibka->Caption = "Незакрытый EB для EI";

if (osh_sozd_shemy == SHEMA_OSH_STEK_FULL)   //
   l_soob_oshibka->Caption = "Начало блока: стек полон";

if (osh_sozd_shemy == SHEMA_OSH_STEK_EMPTY)  //
   l_soob_oshibka->Caption = "Конец блока: стек пуст";

if (osh_sozd_shemy == SHEMA_net_el_wetki)    //
   l_soob_oshibka->Caption = "схема: нет эл. ветки "
   + AnsiString(dalvjaz_tek_wetka+1);

if (osh_sozd_shemy == SHEMA_poterq_el)       //
   l_soob_oshibka->Caption = "схема: потеря элементов";

if (osh_sozd_shemy == SHEMA_net_w_wetki)     //
   l_soob_oshibka->Caption = "схема: нет верт. ветки "
   + AnsiString(dalvjaz_tek_wetka+1);

if (osh_sozd_shemy == SHEMA_poterq_dwert)    //
   l_soob_oshibka->Caption = "схема: потеря вертикалей";

if (osh_sozd_shemy == SHEMA_el_w_steke_LSW)  //
   l_soob_oshibka->Caption = "остались эл. в стеке ЛСВ";

if (osh_sozd_shemy == SHEMA_m_elem_full)     //
   l_soob_oshibka->Caption = "массив элементов полон";

if (osh_sozd_shemy == SHEMA_m_wert_full)     //
   l_soob_oshibka->Caption = "массив вертикалей полон";

if (osh_sozd_shemy == SHEMA_net_gran_el_w)   //
   l_soob_oshibka->Caption = "нет граничных эл.вертикали";

if (osh_sozd_shemy == SHEMA_m_wetok_full)     //
   l_soob_oshibka->Caption = "массив веток полон";
```

```
            if (osh_sozd_shemy == SHEMA_m_otkr_wert_full) //
               l_soob_oshibka->Caption = "мас.откр.верт. полон";

            if (osh_sozd_shemy == SHEMA_no_otkr_wert) //
               l_soob_oshibka->Caption = "не найдена откр.вертикаль";

            if (osh_sozd_shemy == SHEMA_osh_ind_mow)  //
               l_soob_oshibka->Caption = "ошиб.инд.мас.откр.верт.";

            if (osh_sozd_shemy == SHEMA_dmow_full)  //
               l_soob_oshibka->Caption = "доп.мас.откр.верт. полон";


            //-----------------------------------------------


            if (osh_sozd_shemy == FAJL_net_zagol_shemy)
               l_soob_oshibka->Caption = "файл: нет загол. схемы";

            if (osh_sozd_shemy == FAJL_net_zagol_w1)
               l_soob_oshibka->Caption = "файл: нет загол. ветки 1";



            // osh

            tek_str_lsw = tek_el_shemy.n_str;

            for (i=0; i < ListBox1->Items->Count; i++) {
              if (i == tek_str_lsw)
                     ListBox1->Selected[i] = true;
              else   ListBox1->Selected[i] = false;
            }

            l_soob_oshibka->Visible = true;

            wernutx_sostoqnie(Sender);
          }
          else  l_soob_no_osh->Visible = true;
}


// i.12. проверить ошибки структуры

void __fastcall Tform_ls::prow_oshibki_struktury(TObject *Sender)
{
          if (el_steka_lsw.used == IS) {// в стеке есть элементы

            // сравниваем текущий эл-т с взятым из стека


            if ((tek_el_shemy.type == t_EI) &&
                (el_steka_lsw.type == t_EB)) {

              osh_sozd_shemy = SHEMA_EB_for_EI;
              return;
            }

            if ((tek_el_shemy.type == t_EB) &&
                (el_steka_lsw.type == t_EB)) {

              osh_sozd_shemy = SHEMA_powtor_EB;
              return;
            }
```

```
  if ((tek_el_shemy.type == t_CE) &&
      (el_steka_lsw.type == t_IF)) {

    osh_sozd_shemy = SHEMA_IF_for_CE;
    return;
  }

  if ((tek_el_shemy.type == t_CE) &&
      (el_steka_lsw.type == t_EI)) {

    osh_sozd_shemy = SHEMA_EI_for_CE;
    return;
  }

  if ((tek_el_shemy.type == t_CE) &&
      (el_steka_lsw.type == t_EB)) {

    osh_sozd_shemy = SHEMA_EB_for_CE;
    return;
  }

  if ((tek_el_shemy.type == t_EI) &&
      (el_steka_lsw.type == t_CB)) {

    osh_sozd_shemy = SHEMA_CB_for_EI;
    return;
  }

  if ((tek_el_shemy.type == t_EB) &&
      (el_steka_lsw.type == t_CB)) {

    osh_sozd_shemy = SHEMA_CB_for_EB;
    return;
  }

  if ((tek_el_shemy.type == t_E) &&
      (el_steka_lsw.type == t_CB)) {

    osh_sozd_shemy = SHEMA_CB_for_E;
    return;
  }
}
else {// в стеке нет элементов

  if (tek_el_shemy.type == t_EI) {

    osh_sozd_shemy = SHEMA_no_IF_for_EI;
    return;
  }

  if (tek_el_shemy.type == t_EB) {

    osh_sozd_shemy = SHEMA_no_IF_for_EB;
    return;
  }

  if (tek_el_shemy.type == t_E) {

    osh_sozd_shemy = SHEMA_no_IF_for_E;
    return;
  }

  if (tek_el_shemy.type == t_CE) {
```

```
            osh_sozd_shemy = SHEMA_no_CB_for_CE;
            return;
          }
        }
}


// i.14. обработка схемы для  t_IF

void __fastcall Tform_ls::obr_shemy_dlq_IF(TObject *Sender)
{
        obr_shemu_dlq_el(Sender);
        if (osh_sozd_shemy != SHEMA_OK)  return;

        sozdatx_wert(T_WERT_DOCHX);
}


// i.15. создать вертикаль

void __fastcall Tform_ls::sozdatx_wert(int type)
{
/*
             Создание новой дочерней вертикали
           и заполнение массива открытых вертикалей

Дочерняя вертикаль всегда создается для текущей открытой вертикали.

Новая вертикаль всегда ставится на место с n_sprawa на 1 больше,
чем у эл. слева, а ранее открытые вертикали сдвигаются вправо и при
необходимости их n_sprawa последовалельно увеличиваются на 1 слева направо
так, чтобы у каждой открытой вертикали n_sprawa оставался уникальным.
При закрытии вертикали и ее удалении из массива открытых вертикалей
элементы массива открытых вертикалей, находящиеся правее, сдвигаются
влево.
К моменту закрытия вертикали ее n_sprawa определяется окончательно, так
что становится возможным найти ее смещение относительно главной вертикали
ветки по X. В момент закрытия вертикали элементом t_E находим и нижнюю
координату У для этой вертикали.
*/

        int i, k;

        i=KOL_WETOK;
        while (i < MAX_KOL_WERT) {

          if (tek_alg.m_wert[i].used == NO) {

            tek_alg.m_wert[i].wetka = dalvjaz_tek_wetka+1;
            tek_alg.m_wert[i].used = IS;
            tek_alg.m_wert[i].type = type;
            tek_alg.m_wert[i].parent_w = m_otkr_wert[tek_otkr_wert];
            tek_alg.m_wert[i].parent_el = tek_el_shemy.ind_m_el;
            tek_alg.m_wert[i].wet_pel =
                                  tek_alg.m_el[tek_el_shemy.ind_m_el].wetka;
            tek_alg.m_wert[i].n_str_pel = tek_el_shemy.n_str;

            tek_alg.m_el[tek_el_shemy.ind_m_el].n_dwert = i;
              // задаем ссылку текущего элемента на дочернюю вертикаль

            if (type == T_WERT_DOCHX) {
              tek_alg.m_wert[i].y1 =
                        tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].y1 +
```

```
                        ((tek_alg.m_el[tek_el_shemy.ind_m_el].y1 +
                          tek_alg.m_el[tek_el_shemy.ind_m_el].y2)/2);
            }
            if (type == T_WERT_CIKLA) {
              tek_alg.m_wert[i].y1 =
                        tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].y1 +
                          tek_alg.m_el[tek_el_shemy.ind_m_el].y2 +
                          ((dalvjaz_chdy*3)/4);
            }

            tek_alg.m_wert[i].tyel = dalvjaz_chdy + (dalvjaz_chdy/2);

            tek_alg.m_wert[i].type_1_el = -1;

            tek_alg.m_el_wert[i] = 0;

            tek_alg.m_dw_wetki[dalvjaz_tek_wetka]++;

            if (tek_alg.m_inwwet[dalvjaz_tek_wetka] == -1)
                                tek_alg.m_inwwet[dalvjaz_tek_wetka] = i;

            mow__wstawitx(i, tek_otkr_wert+1);
            if (osh_sozd_shemy != SHEMA_OK)   return;

            break;
          }

          i++;
        }

        if (i == MAX_KOL_WERT) {

          osh_sozd_shemy == SHEMA_m_wert_full;
          return;

        }
}


// i.16. обработка схемы для  t_EI

void __fastcall Tform_ls::obr_shemy_dlq_EI(TObject *Sender)
{
        int i, tw;
        // считаем что в стеке t_IF или t_EI - проверка была раньше

        tw = tek_alg.m_el[el_steka_lsw.ind_m_el].n_dwert;

        for (i=0; i < kol_otkr_wert; i++)
          if (m_otkr_wert[i] == tw) {
                                tek_otkr_wert = i;   break;   }

        if (i == kol_otkr_wert) {
          osh_sozd_shemy = SHEMA_no_otkr_wert;
          return;
        }

        obr_shemu_dlq_el(Sender);
        if (osh_sozd_shemy != SHEMA_OK)  return;

        sozdatx_wert(T_WERT_DOCHX);
}
```

```
// i.17. обработка схемы для  t_EB

void __fastcall Tform_ls::obr_shemy_dlq_EB(TObject *Sender)
{
         int i, tw;

         // считаем что в стеке t_IF или t_EI - проверка была раньше

         tw = tek_alg.m_el[el_steka_lsw.ind_m_el].n_dwert;

         for (i=0; i < kol_otkr_wert; i++)
           if (m_otkr_wert[i] == tw) {
                                 tek_otkr_wert = i;   break;   }

         if (i == kol_otkr_wert) {
           osh_sozd_shemy = SHEMA_no_otkr_wert;
           return;
         }

         obr_shemu_dlq_el(Sender);
         if (osh_sozd_shemy != SHEMA_OK)  return;
}


// i.18. обработка для  t_E

void __fastcall Tform_ls::obr_shemy_dlq_E(TObject *Sender)
{
         int i, j, k, dmow[KOL_OTKR_WERT], y, my, kol_dmow;
/*
t_E составляет список всех открытых IF и EI вертикалей и закрывает их

*/
         // ветку t_IF делаем текущей открытой веткой

         // ищем в стеке t_IF

         i=0;
         while (i < kol_s_lsw) {

           give_stek_lsw(i);
           if (el_steka_lsw.type == t_IF)  break;
           i++;
         }
         if (i == kol_s_lsw) {
           osh_sozd_shemy = SHEMA_no_IF_for_E;
           return;
         }

         for (k=0; k < kol_otkr_wert; k++) {
           if (tek_alg.m_el[el_steka_lsw.ind_m_el].n_wert ==
                                          m_otkr_wert[k])  break;  }
         if (k == kol_otkr_wert) {
           osh_sozd_shemy = SHEMA_no_otkr_wert;
           return;
         }

         tek_otkr_wert = k;

         obr_shemu_dlq_el(Sender);
         if (osh_sozd_shemy != SHEMA_OK)  return;


         // загружаем в dmow все вертикали t_IF
```

```
kol_dmow = 0;

for (j = i; j >= 0; j--)  {

  give_stek_lsw(j);

  if (el_steka_lsw.type == t_IF) {

    dmow[kol_dmow] = tek_alg.m_el[el_steka_lsw.ind_m_el].n_wert;
    kol_dmow++;
  }

  dmow[kol_dmow] = tek_alg.m_el[el_steka_lsw.ind_m_el].n_dwert;

  if (kol_dmow == KOL_OTKR_WERT) {

    osh_sozd_shemy = SHEMA_dmow_full;
    return;
  }

  kol_dmow++;
}

// для всех эл. dmow проверяем, а есть ли найденный элемент
// в массиве открытых вертикалей

for (j=0; j < kol_dmow; j++) {

  for (k=0; k < kol_otkr_wert; k++) {

    if (dmow[j] == m_otkr_wert[k])   break;
  }
  if (k == kol_otkr_wert) {
    osh_sozd_shemy = SHEMA_no_otkr_wert;
    return;
  }
}

// вычисляем максимальную y для всех найденных вертикалей

my = 0;

for (j=0; j < kol_dmow; j++) {

  y = tek_alg.m_wert[dmow[j]].y1 + tek_alg.m_wert[dmow[j]].tyel;
  if (j == 0)  y = y - dalvjaz_chdy;
  if (y > my)  my = y;
}

// для всех вертикалей задаем новое значение tyel, задаем ссылку
// на t_E и удаляем их из массива открытых вертикалей

for (j=0; j < kol_dmow; j++) {

  tek_alg.m_wert[dmow[j]].tyel = my - tek_alg.m_wert[dmow[j]].y1;

  tek_alg.m_wert[dmow[j]].parent_endw = m_otkr_wert[tek_otkr_wert];
  tek_alg.m_wert[dmow[j]].parent_end = tek_el_shemy.ind_m_el;
  tek_alg.m_wert[dmow[j]].wet_pend =
                          tek_alg.m_el[tek_el_shemy.ind_m_el].wetka;
  tek_alg.m_wert[dmow[j]].n_str_pend = tek_el_shemy.n_str;

  if (j > 0) {// вертикаль t_IF остается открытой
```

```
            for (k=0; k < kol_otkr_wert; k++) {

              if (dmow[j] == m_otkr_wert[k])   break;
            }

            mow__udalitx(k);
          }
        }

        // корректируем координату y для t_E

        y = tek_alg.m_wert[ tek_alg.m_el[tek_el_shemy.ind_m_el].n_wert ].y1;

        tek_alg.m_el[tek_el_shemy.ind_m_el].y1 = my - y;
        tek_alg.m_el[tek_el_shemy.ind_m_el].y2 = my - y;

        tek_alg.m_wert[ tek_alg.m_el[tek_el_shemy.ind_m_el].n_wert ].tyel =
           tek_alg.m_el[tek_el_shemy.ind_m_el].y1 + dalvjaz_chdy;
}


// i.19. обработка для  t_CB

void __fastcall Tform_ls::obr_shemy_dlq_CB(TObject *Sender)
{
        obr_shemu_dlq_el(Sender);
        if (osh_sozd_shemy != SHEMA_OK)  return;

        sozdatx_wert(T_WERT_CIKLA);
}


// i.20. обработка для  t_CE

void __fastcall Tform_ls::obr_shemy_dlq_CE(TObject *Sender)
{
        int i, tw;

        obr_shemu_dlq_el(Sender);
        if (osh_sozd_shemy != SHEMA_OK)  return;

        // исключаем вертикаль цикла из массива открытых вертикалей
        // задаем ей нижнюю границу и элемент конца

        for (i=0; i < kol_otkr_wert; i++) {

          if (tek_alg.m_el[el_steka_lsw.ind_m_el].n_dwert ==
                              m_otkr_wert[i])  break;
        }
        if (i == kol_otkr_wert) {
          osh_sozd_shemy = SHEMA_no_otkr_wert;
          return;
        }

        tw = m_otkr_wert[i];
        tek_alg.m_wert[tw].parent_endw = m_otkr_wert[tek_otkr_wert];
        tek_alg.m_wert[tw].parent_end = tek_el_shemy.ind_m_el;
        tek_alg.m_wert[tw].wet_pend =
                                    tek_alg.m_el[tek_el_shemy.ind_m_el].wetka;
        tek_alg.m_wert[tw].n_str_pend = tek_el_shemy.n_str;

        tek_alg.m_wert[tw].tyel =
                        ((tek_alg.m_el[tek_el_shemy.ind_m_el].y1 +
```

```
                              tek_alg.m_el[tek_el_shemy.ind_m_el].y2)/2) +
                              tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].y1 -
                              tek_alg.m_wert[tw].y1;
          mow__udalitx(i);
}


// i.21. обработка для  t_HD, t_A, t_AD

void __fastcall Tform_ls::obr_shemu_dlq_HD_A_AD(TObject *Sender)
{
          obr_shemu_dlq_el(Sender);
          if (osh_sozd_shemy != SHEMA_OK)  return;
}


// i.22. обработка схемы для элемента

void __fastcall Tform_ls::obr_shemu_dlq_el(TObject *Sender)
{

          // если новый элемент силуэта, то для него нужно выделить
          // место в массиве элементов

          int i, ii, j, y1;
          uchar  fl_buf;

          if (tek_el_shemy.ind_m_el == -1)  {

            i=0;
            while (i < MAX_KOL_ELEM) {

              if (tek_alg.m_el[i].used == NO)  {

                // поиск в массиве ЛСВ элемента, соответствующего
                // формально пустому месту в tek_alg.m_el[i]

                fl_buf = NO;

                for (j=(tek_el_shemy.n_str+1); j < kol_m_lsw; j++) {

                  if (m_lsw[j].ind_m_el == i) { fl_buf = IS;  break;  }
                }

                if (fl_buf == NO) {

                  // нинциализация нового элемента схемы

                  tek_alg.m_el[i].used    = IS;
                  tek_alg.m_el[i].name[0] = 0;
                  tek_alg.m_el[i].type    = tek_el_shemy.type;
                  tek_alg.m_el[i].n_dwert = -1;
                  tek_alg.m_el[i].n_bmp   = -1;
                  tek_alg.m_el[i].end_w   = NO;

                  tek_el_shemy.ind_m_el   = i;

                  break;
                }
              }

              if (i == MAX_KOL_ELEM) {

                osh_sozd_shemy == SHEMA_m_elem_full;
```

```
      return;
    }

    i++;
  }
}

ii = tek_el_shemy.ind_m_el;

tek_alg.m_el[ii].wetka = dalvjaz_tek_wetka+1;
tek_alg.m_el[ii].used = IS;
tek_alg.m_el[ii].old = NO;
tek_alg.m_el[ii].n_str  = tek_el_shemy.n_str;
tek_alg.m_el[ii].n_wert = m_otkr_wert[tek_otkr_wert];

tek_alg.m_el[ii].end_w = NO;

tek_alg.m_el_wetki[ dalvjaz_tek_wetka ]++;

tek_alg.m_el_wert[tek_alg.m_el[ii].n_wert]++;

tek_alg.m_el[ii].n_na_wert =
                      tek_alg.m_el_wert[tek_alg.m_el[ii].n_wert];

if (tek_alg.m_el_wert[tek_alg.m_el[ii].n_wert] == 1) {

  //tek_alg.m_inew[tek_alg.m_el[ii].n_wert] = ii;
  // m_inew придется искать снова в конце создания схемы после
  // сортировки элементов по порядковому номеру ветки

  tek_alg.m_wert[tek_alg.m_el[ii].n_wert].type_1_el =
                                              tek_el_shemy.type;
}

// находим координаты эл-та на вертикали

tek_alg.m_el[ii].x1 = -3 * dalvjaz_chw;
tek_alg.m_el[ii].x2 = 3 * dalvjaz_chw;

y1 = tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].tyel;
tek_alg.m_el[ii].y1 = y1;

if (tek_alg.m_el[ii].type == t_EB) {
  tek_alg.m_el[ii].y1 = 0;   tek_alg.m_el[ii].y2 = 0;
  // ELSE помещаем на верхнюю вершину порожденной вертикали
}
else
if (tek_alg.m_el[ii].type != t_E)
      tek_alg.m_el[ii].y2 = y1+((3*dalvjaz_chdy)/2);
else  tek_alg.m_el[ii].y2 = y1;

//if (tek_alg.m_el[ii].type == t_E)  y1 = tek_alg.m_el[ii].y2;
//else {
  y1 = tek_alg.m_el[ii].y2 + dalvjaz_chdy;
  if ((tek_alg.m_el[ii].type == t_IF) ||
      (tek_alg.m_el[ii].type == t_EI) ||
      (tek_alg.m_el[ii].type == t_CB) ||
      (tek_alg.m_el[ii].type == t_CE))  y1 += dalvjaz_chdy / 2;
//}
tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].tyel = y1;

if ((y1+tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].y1) > dalvjaz_ymax)

   dalvjaz_ymax = y1+tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].y1;
```

```
}


// i.23. функция сортировки "меньше" для массива элементов

int less_m_el(int i, int j)
{
  return (tek_alg.m_el[mie_wetki[i]].n_str < tek_alg.m_el[mie_wetki[j]].n_str);
}


// i.24. функция сортировки "поменять местами" для массива элементов

void  swap_m_el(int i, int j)
{
  t_elem w;

  memcpy(&w, &tek_alg.m_el[mie_wetki[i]], sizeof(t_elem));
  memcpy(&tek_alg.m_el[mie_wetki[i]],
         &tek_alg.m_el[mie_wetki[j]], sizeof(t_elem));
  memcpy(&tek_alg.m_el[mie_wetki[j]], &w, sizeof(t_elem));
}


// i.25. обработка после окончания сканирования ЛСВ

void __fastcall Tform_ls::obr_shemy_w_konce_lsw(TObject *Sender)
{
         int i, j, k, ti;

         // удаление элементов текущей ветки, которые остались старыми
         // не привязанными к вертикалям

         i = 0;

         while (i < MAX_KOL_ELEM) {

           if (tek_alg.m_el[i].used == IS)  {

             if (tek_alg.m_el[i].wetka == (dalvjaz_tek_wetka+1))  {

               if (tek_alg.m_el[i].old == IS)  tek_alg.m_el[i].used = NO;
             }
           }

           i++;
         }

         // новое число элементов схемы

         tek_alg.kol_elem = tek_alg.kol_elem -
                            tek_alg.m_old_el_wetki[dalvjaz_tek_wetka] +
                            tek_alg.m_el_wetki[dalvjaz_tek_wetka];

         // находим индексы элементов ветки

         j = 0;  i = 0;
         while (i < MAX_KOL_ELEM) {

           if (tek_alg.m_el[i].used == IS)  {

             if (tek_alg.m_el[i].wetka == (dalvjaz_tek_wetka+1))  {
```

```
      mie_wetki[j] = i;   j++;   }
  }

  i++;
}

// сортируем элементы ветки по порядку следования в ЛСВ

Sort(0, tek_alg.m_el_wetki[dalvjaz_tek_wetka]-1,
                                /*&*/less_m_el, /*&*/swap_m_el);

// новый индекс начального элемента ветки

tek_alg.m_inewet[dalvjaz_tek_wetka] = mie_wetki[0];


// находим для дополнительных вертикалей ветки m_inew

i=0;
while (i < MAX_KOL_ELEM) {

  if ((tek_alg.m_el[i].used == IS) &&
      (tek_alg.m_el[i].wetka == (dalvjaz_tek_wetka+1))) {

      ti = tek_alg.m_el[i].n_wert;
      if (tek_alg.m_inew[ti] == -1)  tek_alg.m_inew[ti] = i;
  }
  i++;
}


// восстанавливаем для вертикалей ветки ссылки на родительские
// элементы и на элементы конца ветки

j = KOL_WETOK;
while (j < MAX_KOL_WERT) {

  // ищем элементы с нужными для родительского элемента
  // вертикали и конца вертикали номерами веток и строк ЛСВ

  if ((tek_alg.m_wert[j].wetka == dalvjaz_tek_wetka+1) &&
      (tek_alg.m_wert[j].used == IS))  {

    tek_alg.m_wert[j].parent_el = -1;
    tek_alg.m_wert[j].parent_end = -1;

    i = tek_alg.m_inew[j];
    while (i < MAX_KOL_ELEM) {

      if ((tek_alg.m_el[i].wetka == tek_alg.m_wert[j].wet_pel) &&
          (tek_alg.m_el[i].n_str == tek_alg.m_wert[j].n_str_pel)) {

                        tek_alg.m_wert[j].parent_el = i;  }

      if ((tek_alg.m_el[i].wetka == tek_alg.m_wert[j].wet_pend) &&
          (tek_alg.m_el[i].n_str == tek_alg.m_wert[j].n_str_pend)) {

                        tek_alg.m_wert[j].parent_end = i;  }

      if ((tek_alg.m_wert[j].parent_el != -1) &&
          (tek_alg.m_wert[j].parent_end != -1))  break;
      i++;
    }
    if (i == MAX_KOL_ELEM) {
```

```
            osh_sozd_shemy = SHEMA_net_gran_el_w;
            return;
          }
        }

        j++;
      }

      // присваиваем элементам схемы новые порядковые номера

      ti = tek_alg.m_el[ mie_wetki[0] ].ind;

      for (j=dalvjaz_tek_wetka; j < (tek_alg.kol_wetok-1); j++) {

        i= tek_alg.m_inewet[j];
        while (i < MAX_KOL_ELEM) {

          if (tek_alg.m_el[i].used == IS)  {

            if ((tek_alg.m_el[i].wetka == j+1) &&
                (tek_alg.m_el[i].type != t_E) &&
                (tek_alg.m_el[i].type != t_EB))  {

              tek_alg.m_el[i].ind = ti;   ti++;   }
          }

          i++;
        }
      }
}


// i.26. запомнить состояние схемы

void __fastcall Tform_ls::zapomnitx_sostoqnie(TObject *Sender)
{
      if (fl_otkata_shemy == IS)  return;

      fl_otkata_shemy = IS;
      form_dalvjaz->b_zapisxClick(Sender);
      fl_otkata_shemy = NO;
}


// i.27. вернуть состояние схемы

void __fastcall Tform_ls::wernutx_sostoqnie(TObject *Sender)
{
      int tx, dtw, i, j, n;
      char stro[dl_stro], st[50];

      if (fl_otkata_shemy == IS)  return;

      dtw = dalvjaz_tek_wetka;
      n = tek_el_shemy.n_str;

      fl_otkata_shemy = IS;
      form_dalvjaz->b_chtenieClick(Sender);
      fl_otkata_shemy = NO;

      l_soob_no_osh->Visible = false;

      dalvjaz_tek_wetka = dtw;
```

```
tx = tek_alg.m_x_wetok[dalvjaz_tek_wetka] - (dalvjaz_chw*5);

form_dalvjaz->ScrollBar2->Position = (tx*100)/dalvjaz_xmax;


zapisx_LSW_iz_shemy(Sender);

kol_m_lsw2 = 0;

for (i=0; i < ListBox1->Items->Count; i++) {

  strcpy(stro, ListBox1->Items->Strings[i].c_str());
  strncpy(st, stro, 3);   st[3]=0;

  if (strstr(stro, "IF") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_IF;
  }
  else
  if (strstr(stro, "EI") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_EI;
  }
  else
  if (strstr(stro, "EB") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_EB;
  }
  else
  if (strstr(stro, "CB") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_CB;
  }
  else
  if (strstr(stro, "CE") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_CE;
  }
  else
  if (strstr(stro, "HD") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_HD;
  }
  else
  if (strstr(stro, "AD") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_AD;
  }
  else
  if (strstr(stro, "E ") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_E;
  }
  else
  if (strstr(stro, "A ") != NULL) {

    m_lsw2[kol_m_lsw2].type = t_A;
  }
  else  break;  // пошли пустые строки

  m_lsw2[kol_m_lsw2].ind = atoi(st);
  m_lsw2[kol_m_lsw2].ind_m_el = -1;
```

```
          kol_m_lsw2++;
        }

        // сравниваем с прочитанным m_lsw - оттуда, где совпадают
        // индексы на экране, берем индексы элементов в m_el

        for (i=0; i < kol_m_lsw2; i++) {

          for (j=0; j < kol_m_lsw; j++) {

            if (m_lsw2[i].ind == m_lsw[j].ind) {
                 m_lsw2[i].ind_m_el = m_lsw[j].ind_m_el;
                 break;  }
          }
        }

        memcpy(&m_lsw[0], &m_lsw2[0], sizeof(el_lsw)*RAZM_LSW);
        kol_m_lsw = kol_m_lsw2;

        chtenie_LSW(Sender);

        if ((n >= 0) && (n < ListBox1->Items->Count))
                                  ListBox1->Selected[n] = true;
        osh_sozd_shemy = SHEMA_OK;
}


// i.28.

//---------------------------------------------------------------------------

// i.29. обработка для кнопки "откат схемы"

void __fastcall Tform_ls::b_otmena_shemyClick(TObject *Sender)
{
        wernutx_sostoqnie(Sender);
}
//---------------------------------------------------------------------------

// i.30. запись ЛСВ из схемы

void __fastcall Tform_ls::zapisx_LSW_iz_shemy(TObject *Sender)
{
        int j, ii;


        kol_stotm = 0;  // сброс стека отмены ЛСВ
        //kol_b_lsw = 0;  // сброс буфера переноса ЛСВ


        kol_m_lsw = tek_alg.m_el_wetki[dalvjaz_tek_wetka];

        ii = tek_alg.m_inewet[dalvjaz_tek_wetka];

        j=0;
        while ((j < tek_alg.m_el_wetki[dalvjaz_tek_wetka]) &&
                                                 (ii < MAX_KOL_ELEM)) {

          if ((tek_alg.m_el[ii].used == IS) &&
              (tek_alg.m_el[ii].wetka == (dalvjaz_tek_wetka+1))) {

            m_lsw[j].ind      = tek_alg.m_el[ii].ind;
            m_lsw[j].type     = tek_alg.m_el[ii].type;
            m_lsw[j].ind_m_el = ii;
```

```
            j++;
          }

          ii++;
        }
}


// i.31. чтение ЛСВ

void __fastcall Tform_ls::chtenie_LSW(TObject *Sender)
{
        int j, ii, n, len, tp,
            tl=1; // текущая линия ЛСВ
        char stro[dl_stro], st[50];

        if (fl_otkata_shemy == IS)   return;

        ListBox1->Clear();

        for (j=0; j < kol_m_lsw; j++) {

          if (m_lsw[j].ind == 0) {

            if (m_lsw[j].ind_m_el == -1)    strcpy(stro, "    ");
            else                            strcpy(stro, "*   ");
          }
          else {

            itoa(m_lsw[j].ind, stro, 10);

            len = strlen(stro);

            while (len < 4)  {  strcat(stro, " ");  len++;  }
          }

          strcat(stro, "|  |  |  |  |  |  |  |");

          if (m_lsw[j].type == t_IF)  strcpy(st, "IF");
          if (m_lsw[j].type == t_EI)  strcpy(st, "EI");
          if (m_lsw[j].type == t_EB)  strcpy(st, "EB");
          if (m_lsw[j].type == t_E)   strcpy(st, "E ");
          if (m_lsw[j].type == t_CB)  strcpy(st, "CB");
          if (m_lsw[j].type == t_CE)  strcpy(st, "CE");
          if (m_lsw[j].type == t_HD)  strcpy(st, "HD");
          if (m_lsw[j].type == t_A)   strcpy(st, "A ");
          if (m_lsw[j].type == t_AD)  strcpy(st, "AD");

          if ((m_lsw[j].type == t_EI) ||
              (m_lsw[j].type == t_EB) ||
              (m_lsw[j].type == t_E)  ||
              (m_lsw[j].type == t_CE))   tl--;

          if (tl < 1)  tl = 1;

          tp = (tl*3)+2;   stro[tp] = st[0];   stro[tp+1] = st[1];

          if ((m_lsw[j].type == t_IF) ||
              (m_lsw[j].type == t_EI) ||
              (m_lsw[j].type == t_EB) ||
              (m_lsw[j].type == t_CB))   tl++;

          if (tl > 7)  tl = 7;
```

```
          ListBox1->Items->Add(AnsiString(stro));
        }

        n = (ListBox1->Height / ListBox1->ItemHeight);

        ii = n - kol_m_lsw;

        if (ii > 0) {

          for (j=0; j < ii; j++)
              ListBox1->Items->Add("    |  |  |  |  |  |  |  |");
        }
        else  ListBox1->Items->Add("    |  |  |  |  |  |  |  |");

}



// i.32. проверка буфера ЛСВ при замене ветки

void  prow_buf_lsw_pri_smene_wetki()
{
        int i;

        if (kol_b_lsw == 0)  return;

        for (i=0; i < kol_b_lsw; i++) {

          if (buf_lsw[i].ind_m_el != -1) {

            // не должно быть ссылок из буфера на активный эл. схемы

            if (tek_alg.m_el[buf_lsw[i].ind_m_el].used == IS)
                                                  buf_lsw[i].ind_m_el = -1;
          }
        }
}



// i.33. загрузка ЛСВ

void __fastcall Tform_ls::zagruzka_LSW(TObject *Sender)
{
        int i;
        char stro[dl_stro];

        prow_buf_lsw_pri_smene_wetki();

        l_n_wetki->Caption = AnsiString(dalvjaz_tek_wetka+1)+":";

        i = tek_alg.m_inewet[dalvjaz_tek_wetka];
        if (tek_alg.m_el[i].kol_s_k != 0) {
           strcpy(stro, tek_alg.m_el[i].m_k[0].m);
           stro[23] = 0;
           l_z_n_wetki->Caption = AnsiString(stro);
        }
        else l_z_n_wetki->Caption = AnsiString("Пустой заголовок");

        zapisx_LSW_iz_shemy(Sender);
        chtenie_LSW(Sender);
}
```

```
// i.34. поместить элемент в стек ЛСВ

void  push_stek_lsw()
{
         int i;

         if (kol_s_lsw == RAZM_STEKA_LSW) {

           form_ls->osh_sozd_shemy = SHEMA_OSH_STEK_FULL;

           return;
         }

         for (i=kol_s_lsw; i > 0; i--)
           memcpy(&stek_lsw[i], &stek_lsw[i-1], sizeof(el_lsw));

         memcpy(&stek_lsw[0], &tek_el_shemy, sizeof(el_lsw));

         stek_lsw[0].used = IS;

         kol_s_lsw++;
}


// i.35. получить элемент из стека ЛСВ

void  pop_stek_lsw()
{
         int i;

         if (kol_s_lsw == 0) {

           form_ls->osh_sozd_shemy = SHEMA_OSH_STEK_EMPTY;

           return;
         }

         memcpy(&el_steka_lsw, &stek_lsw[0], sizeof(el_lsw));

         for (i=0; i < (kol_s_lsw-1); i++) {

           memcpy(&stek_lsw[i], &stek_lsw[i+1], sizeof(el_lsw));
         }

         stek_lsw[kol_s_lsw-1].used = NO;

         kol_s_lsw--;
}


// i.36. получить запрашиваемый элемент стека ЛСВ

void  give_stek_lsw(int n)
{
         memcpy(&el_steka_lsw, &stek_lsw[n], sizeof(el_lsw));

}



// i.37. сброс стека ЛСВ

void  sbros_steka_lsw()
{
```

```
        int i;

        kol_s_lsw = 0;

        for (i=0; i < RAZM_STEKA_LSW; i++)  stek_lsw[i].used = NO;
}



// i.38. действия при активации формы ЛСВ

void __fastcall Tform_ls::FormActivate(TObject *Sender)
{
        Timer1->Enabled = true;
}
//---------------------------------------------------------------------------

// i.39. действия при деактивации формы ЛСВ

void __fastcall Tform_ls::FormDeactivate(TObject *Sender)
{
        l_soob->Visible = false;
}
//---------------------------------------------------------------------------

// i.40. обработка щелчка мыши по форме ЛСВ

void __fastcall Tform_ls::FormClick(TObject *Sender)
{
        l_soob_oshibka->Visible = false;
        l_soob_no_osh->Visible = false;
}
//---------------------------------------------------------------------------

// i.41. обработка щелчка мыши по списку ЛСВ

void __fastcall Tform_ls::ListBox1Click(TObject *Sender)
{
        l_soob_oshibka->Visible = false;
        l_soob_no_osh->Visible = false;
}
//---------------------------------------------------------------------------

// i.42. инициализация массива открытых вертикалей

void  mow__init(int tw)
{
        m_otkr_wert[0] = tw;
        kol_otkr_wert = 1;
        tek_otkr_wert = 0;
}


// i.43. получить для запрашиваемой открытой вертикали
//       индекс в массиве вертикалей схемы

int  mow__n_wert(int n)
{
        if ((n < 0) || (n >= kol_otkr_wert)) {

          form_ls->osh_sozd_shemy = SHEMA_osh_ind_mow;
          return -1;
        }
```

```
        return  m_otkr_wert[n];
}


// i.44. вставить верт. с индексом tw на n-е место
//       массива открытых вертикалей

void  mow__wstawitx(int tw, int n)
{
        int nn, i;

        if ((n < 1) || (n > MAX_OTKR_WERT)) {

          form_ls->osh_sozd_shemy = SHEMA_osh_ind_mow;
          return;
        }

        nn = n;

        // если индекс больше числа эл. мас.откр.верт., вставляем инд.верт.
        // в конец массива

        if (nn > kol_otkr_wert)  nn = kol_otkr_wert;

        if (kol_otkr_wert == KOL_OTKR_WERT) {

          form_ls->osh_sozd_shemy = SHEMA_m_otkr_wert_full;
          return;
        }

        kol_otkr_wert++;

        for (i=(kol_otkr_wert-1); i > nn; i--)
                                    m_otkr_wert[i] = m_otkr_wert[i-1];
        m_otkr_wert[nn] = tw;

        tek_alg.m_wert[tw].n_sprawa =
                          tek_alg.m_wert[m_otkr_wert[nn-1]].n_sprawa+1;

        // увеличиваем n_sprawa для эл-тов правее, если нужно

        for (i=nn; i < (kol_otkr_wert-1); i++) {

          if (tek_alg.m_wert[m_otkr_wert[i]].n_sprawa ==
              tek_alg.m_wert[m_otkr_wert[i+1]].n_sprawa)
                           tek_alg.m_wert[m_otkr_wert[i+1]].n_sprawa++;
        }
}


// i.45 удалить вертикаль из массива открытых вертикалей

void  mow__udalitx(int n)
{
        int i;

        if ((n < 1) || (n >= kol_otkr_wert)) {

          form_ls->osh_sozd_shemy = SHEMA_osh_ind_mow;
          return;
        }

        for (i=n; i < (kol_otkr_wert-1); i++)
                                  m_otkr_wert[i] = m_otkr_wert[i+1];
```

```
         kol_otkr_wert--;
}


// i.46 преобразовать тип элемента в строку

void  el_type_to_str(int type, char *stro)
{
         if (type == t_IF)
           strcpy(stro, "IF :  блок условия");
         else
         if (type == t_EI)
           strcpy(stro, "EI - ELSE IF :  дополнительный блок условия");
         else
         if (type == t_EB)
           strcpy(stro, "EB - ELSE :  альтернативный блок условия");
         else
         if (type == t_E)
           strcpy(stro, "E - END :  конец блока условия");
         else
         if (type == t_CB)
           strcpy(stro, "CB :  начало цикла");
         else
         if (type == t_CE)
           strcpy(stro, "CE :  конец цикла");
         else
         if (type == t_HD)
           strcpy(stro, "HD :  заголовок ветки");
         else
         if (type == t_AD)
           strcpy(stro, "AD :  адрес перехода к ветке");
         else
         if (type == t_A)
           strcpy(stro, "A - action :  действие");
         else
         if (type == t_RET)
           strcpy(stro, "RET - RETURN :  возврат из процедуры");
         else
           strcpy(stro, "неизвестный тип элемента");
}


// i.47. проверка является ли строка пустой или строкой пробелов

uchar  str_probelow(char *str)
{
         int l,n;

         l = strlen(str);
         n=0;   while (str[n] == ' ')  n++;

         if (n == l)  return  IS;
         else         return  NO;
}


// i.48. заготовка для пересчета координат элементов после
//       завершения построения схемы

void __fastcall Tform_ls::pereschet_koord_el(TObject *Sender)
{
/*=======================
 старый   podschet_koordinat  для веток
```

```
          int j, i, n, kol, x1, y1, x2, y2;

          dalvjaz_ymax = 0;

          for (j=0; j < tek_alg.kol_wetok; j++) {

            i = tek_alg.m_inewet[j];   kol = tek_alg.m_el_wetki[j];

            y1 = 0;

            n=0;
            while (n < kol) {

              if (tek_alg.m_el[i].wetka == (j+1)) {

                if ((tek_alg.m_el[i].used == IS) || (i >= MAX_KOL_ELEM)) {

                  tek_alg.m_el[i].x1 = -3 * dalvjaz_chw;
                  tek_alg.m_el[i].x2 = 3 * dalvjaz_chw;
                  tek_alg.m_el[i].y1 = y1;
                  tek_alg.m_el[i].y2 = y1+((3*dalvjaz_chdy)/2);

                  y1 = tek_alg.m_el[i].y2 + dalvjaz_chdy;
                  if ((tek_alg.m_el[i].type == t_IF) ||
                      (tek_alg.m_el[i].type == t_EI))
                                              y1 += dalvjaz_chdy / 2;
                  if (y1 > dalvjaz_ymax) dalvjaz_ymax = y1;

                  n++;
                }
              }

              i++;
            }
          }


//============================================================
// obr_shemu_dlq_el


          // находим координаты эл-та на вертикали

          tek_alg.m_el[ii].x1 = -3 * dalvjaz_chw;
          tek_alg.m_el[ii].x2 = 3 * dalvjaz_chw;

          y1 = tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].tyel;
          tek_alg.m_el[ii].y1 = y1;

          if (tek_alg.m_el[ii].type == t_EB) {
            tek_alg.m_el[ii].y1 = 0;   tek_alg.m_el[ii].y2 = 0;
            // ELSE помещаем на верхнюю вершину порожденной вертикали
          }
          else
          if (tek_alg.m_el[ii].type != t_E)
                tek_alg.m_el[ii].y2 = y1+((3*dalvjaz_chdy)/2);
          else  tek_alg.m_el[ii].y2 = y1;

          //if (tek_alg.m_el[ii].type == t_E)  y1 = tek_alg.m_el[ii].y2;
          //else {
            y1 = tek_alg.m_el[ii].y2 + dalvjaz_chdy;
            if ((tek_alg.m_el[ii].type == t_IF) ||
                (tek_alg.m_el[ii].type == t_EI) ||
                (tek_alg.m_el[ii].type == t_CB) ||
```

```
            (tek_alg.m_el[ii].type == t_CE))  y1 += dalvjaz_chdy / 2;
        //}
        tek_alg.m_wert[m_otkr_wert[tek_otkr_wert]].tyel = y1;

*/

}
```

ПРИЛОЖЕНИЕ С. Модуль U_DALVJ – обработка ВС в dal_vjaz

Файл U_DALVJ.H

```
//---------------------------------------------------------------------------
#ifndef gbsH
#define gbsH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <ComCtrls.hpp>
#include <Menus.hpp>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//---------------------------------------------------------------------------
class Tform_dalvjaz : public TForm
{
__published:      // IDE-managed Components
    TScrollBar *ScrollBar1;
    TPanel *Panel1;
    TScrollBar *ScrollBar2;
    TPanel *Panel2;
    TPanel *Panel3;
    TButton *b_about;
    TPaintBox *PaintBox1;
    TTabControl *TabControl1;
    TButton *b_chtenie;
    TButton *b_zapisx;
    TLabel *Label1;
    TButton *b_bs_n;
    TButton *b_bs_min;
    TButton *b_bs_pl;
    TButton *b_bs_k;
    TLabel *Label2;
    TButton *b_w_n;
    TButton *b_w_min;
    TButton *b_w_pl;
    TButton *b_w_k;
    TComboBox *ComboBox1;
    TPopupMenu *PopupMenu1;
    TButton *b_bmp;
    TPanel *Panel4;
    TImage *Image1;
    TMenuItem *mn_danet;
    TMenuItem *mn_red;
    TMenuItem *N1;
    TMenuItem *mn_copy;
    TMenuItem *mn_calc;
    TMenuItem *mn_data;
    TMenuItem *mn_ekran;
    TMenuItem *mn_err;
    TMenuItem *mn_iserr;
    TMenuItem *mn_otkr;
    TMenuItem *mn_prowdt;
    TMenuItem *mn_recdt;
    TMenuItem *mn_zakr;
    TMenuItem *mn_net_znachka;
```

```
    void __fastcall FormCreate(TObject *Sender);
    void __fastcall ScrollBar1Change(TObject *Sender);
    void __fastcall FormResize(TObject *Sender);
    void __fastcall b_aboutClick(TObject *Sender);


    void __fastcall PaintBox1Paint(TObject *Sender);
    void __fastcall FormDestroy(TObject *Sender);
    void __fastcall PaintBox1MouseDown(TObject *Sender,
          TMouseButton Button, TShiftState Shift, int X, int Y);
    void __fastcall b_bmpClick(TObject *Sender);
    void __fastcall ScrollBar2Change(TObject *Sender);
    void __fastcall b_w_nClick(TObject *Sender);
    void __fastcall b_w_kClick(TObject *Sender);
    void __fastcall b_w_minClick(TObject *Sender);
    void __fastcall b_w_plClick(TObject *Sender);

    void __fastcall PaintBox1MouseMove(TObject *Sender, TShiftState Shift,
          int X, int Y);

    void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
    void __fastcall b_chtenieClick(TObject *Sender);
    void __fastcall b_zapisxClick(TObject *Sender);
    void __fastcall mn_danetClick(TObject *Sender);
    void __fastcall mn_redClick(TObject *Sender);
    void __fastcall mn_copyClick(TObject *Sender);
    void __fastcall mn_calcClick(TObject *Sender);
    void __fastcall mn_dataClick(TObject *Sender);
    void __fastcall mn_ekranClick(TObject *Sender);
    void __fastcall mn_errClick(TObject *Sender);
    void __fastcall mn_iserrClick(TObject *Sender);
    void __fastcall mn_otkrClick(TObject *Sender);
    void __fastcall mn_prowdtClick(TObject *Sender);
    void __fastcall mn_recdtClick(TObject *Sender);
    void __fastcall mn_zakrClick(TObject *Sender);
    void __fastcall mn_net_znachkaClick(TObject *Sender);
private:    // User declarations
public:           // User declarations
    __fastcall Tform_dalvjaz(TComponent* Owner);

    int wspl_inf_okno; // для wsplyw_inf_okno
    int wspl_fl_mouse; // для wsplyw_inf_okno
    int mouse_x, mouse_y, mouse_ind_el;

    void __fastcall podschet_max_koord(TObject *Sender);
    void __fastcall wywod_shemy(TObject *Sender);
    void __fastcall wywod_elem(TObject *Sender, int w, int ie, int n);
    void __fastcall wsplyw_inf_okno(int X, int Y, int j, int ii);
    void __fastcall read_comment(char *fn);
    void __fastcall write_comment(char *fn);

};
//---------------------------------------------------------------------------
extern PACKAGE Tform_dalvjaz *form_dalvjaz;
//---------------------------------------------------------------------------


#endif
```

Файл U_DALVJ.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include <process.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mem.h>
#include <dir.h>
#include "u_dalvj.h"
#include "u_ls.h"
#include "u_red.h"
#include "about.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
Tform_dalvjaz *form_dalvjaz;


//---------------------------------------------------------------------------

Graphics::TBitmap *m_Bitmap[RAZM_M_BITMAP];


//---------------------------------------------------------------------------



// i.2. вывод линии для ВС

void  dalvjaz_line(int x1, int y1, int x2, int y2)
{
         form_dalvjaz->PaintBox1->Canvas->MoveTo(x1,y1);
         form_dalvjaz->PaintBox1->Canvas->LineTo(x2,y2);
         if (x1 != x2)
           form_dalvjaz->PaintBox1->Canvas->LineTo(x2+1,y2);
}


// i.3. вывод текста для ВС

void  dalvjaz_str(int x, int y, char* stro)
{
         form_dalvjaz->PaintBox1->Canvas->TextOut(x,y, stro);
}


// i.4. задание размера фонта для ВС

void  dalvjaz_font(int n)
{
         form_dalvjaz->PaintBox1->Canvas->Font->Size = n;
}

//---------------------------------------------------------------------------
// i.5. конструктор формы ВС

__fastcall Tform_dalvjaz::Tform_dalvjaz(TComponent* Owner)
    : TForm(Owner)
```

```
{
}

//---------------------------------------------------------------------------

// i.6. обработка для создания формы ВС

void __fastcall Tform_dalvjaz::FormCreate(TObject *Sender)
{
          int i;
          FILE *fl;
          char stro[200];
          int f_l, ft, fw, fh;


          wspl_fl_mouse = NO;


          fl = fopen("formsize.dat", "rt");
          if (fl != NULL) {

            if (fgets(stro, 199, fl) != NULL)  {

              sscanf(stro, "%d  %d %d %d", &f_l, &ft, &fw, &fh);

              fclose(fl);

              Left = f_l;    Top = ft;     Width = fw;    Height = fh;
            }
          }


          dalvjaz_beg = 1;


          for (i=0; i < RAZM_M_BITMAP; i++)
                       m_Bitmap[i] = new Graphics::TBitmap;

          // рисунки имеют размеры (21 по x) * (20 по y)

          m_Bitmap[0]->LoadFromFile("p_copy.bmp");
          strcpy(m_imen_bmp[0], "p_copy.bmp");

          m_Bitmap[1]->LoadFromFile("p_culc.bmp");
          strcpy(m_imen_bmp[1], "p_culc.bmp");

          m_Bitmap[2]->LoadFromFile("p_data.bmp");
          strcpy(m_imen_bmp[2], "p_data.bmp");

          m_Bitmap[3]->LoadFromFile("p_ekran.bmp");
          strcpy(m_imen_bmp[3], "p_ekran.bmp");

          m_Bitmap[4]->LoadFromFile("p_err.bmp");
          strcpy(m_imen_bmp[4], "p_err.bmp");

          m_Bitmap[5]->LoadFromFile("p_iserr.bmp");
          strcpy(m_imen_bmp[5], "p_iserr.bmp");

          m_Bitmap[6]->LoadFromFile("p_otkr.bmp");
          strcpy(m_imen_bmp[6], "p_otkr.bmp");

          m_Bitmap[7]->LoadFromFile("p_prowdt.bmp");
          strcpy(m_imen_bmp[7], "p_prowdt.bmp");
```

```
        m_Bitmap[8]->LoadFromFile("p_recdt.bmp");
        strcpy(m_imen_bmp[8], "p_recdt.bmp");

        m_Bitmap[9]->LoadFromFile("p_zakr.bmp");
        strcpy(m_imen_bmp[9], "p_zakr.bmp");

}
//---------------------------------------------------------------------------

// i.7. обработка для вертикальной полосы прокрутки ВС

void __fastcall Tform_dalvjaz::ScrollBar1Change(TObject *Sender)
{
        int pos, smyel;

        pos = ScrollBar1->Position;

        smyel = DALVJAZ_Y_ZAGOL +((3*dalvjaz_chdy)/2) + dalvjaz_chdy +
                                                        ((3*dalvjaz_chdy)/2);

        dalvjaz_smy = -((pos*(dalvjaz_ymax+smyel))/100);

        wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.8. обработка для изменения размеров формы ВС

void __fastcall Tform_dalvjaz::FormResize(TObject *Sender)
{
        Panel1->Width      = Width - 9;
        b_bmp->Left        = Panel1->Width - 81+24;
        ScrollBar1->Left   = Width - 26;
        PaintBox1->Width   = Width - 25;
        Panel2->Left       = Width - 26;
        ScrollBar2->Width  = Width - 26;
        Panel3->Width      = Width - 9;
        b_about->Left      = Panel3->Width - 57;
        TabControl1->Width = Width - 10;

        PaintBox1->Height  = Height - 125;
        ScrollBar2->Top    = Height - 125;
        Panel2->Top        = Height - 70;
        ScrollBar1->Height = Height - 125;
        ScrollBar2->Top    = Height - 70;
        Panel3->Top        = Height - 52;

        form_ls->Top       = Top;
        form_ls->Left      = Left - form_ls->Width;
}
//---------------------------------------------------------------------------

// i.9. обработка для кнопки "о программе"

void __fastcall Tform_dalvjaz::b_aboutClick(TObject *Sender)
{
        AboutBox->Show();
}
//---------------------------------------------------------------------------

// i.10. обработка перерисовки ВС

void __fastcall Tform_dalvjaz::PaintBox1Paint(TObject *Sender)
{
```

```
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.11. обработка при удалении формы

void __fastcall Tform_dalvjaz::FormDestroy(TObject *Sender)
{
         FILE *fl;
         int i;


         if (WindowState != wsMaximized) {

           fl = fopen("formsize.dat", "wt");

           fprintf(fl, "%d %d %d %d\n", Left, Top, Width, Height);

           fclose(fl);
         }


         for (i=0; i < RAZM_M_BITMAP; i++)  delete m_Bitmap[i];

}
//---------------------------------------------------------------------------

// i.12. обработка при щелчке мыши по ВС

void __fastcall Tform_dalvjaz::PaintBox1MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
         int j, i, ii, k, x1, x2, y1, y2;
         tagPOINT p1, p2;


         if (Button == mbRight)  {//

         for (j=0; j < (tek_alg.kol_wetok-1); j++) {

           ii = tek_alg.m_inew[j];

           i=0;
           while (i < tek_alg.m_el_wert[j]) {

             if (tek_alg.m_el[ii].n_wert == j) {

               if ((tek_alg.m_el[ii].used == IS) ||
                   (j == (tek_alg.kol_wetok-1))) {

                 wspl_fl_mouse = IS;
                 wsplyw_inf_okno(X, Y, j, ii);
                 wspl_fl_mouse = NO;
                 if (mouse_ind_el != -1) {

                   p1.x = X;   p1.y = Y;
                   p2 = PaintBox1->ClientToScreen(p1);
                   PopupMenu1->Popup(p2.x, p2.y);
                   return;
                 }
                 i++;
               }
             }
             ii++;
```

```
          }
        }////

        // сканирование элементов дополнительных вертикалей веток

        for (j=0; j < tek_alg.kol_dwert; j++) {////


          if (tek_alg.m_el_wert[KOL_WETOK+j] != 0) {////

          ii = tek_alg.m_inew[KOL_WETOK+j];

          i=0;
          while (i < tek_alg.m_el_wert[KOL_WETOK+j]) {

            if (tek_alg.m_el[ii].n_wert == (KOL_WETOK+j)) {

              if (tek_alg.m_el[ii].used == IS) {

                wspl_fl_mouse = IS;
                wsplyw_inf_okno(X, Y, KOL_WETOK+j, ii);
                wspl_fl_mouse = NO;
                if (mouse_ind_el != -1) {

                  p1.x = X;   p1.y = Y;
                  p2 = PaintBox1->ClientToScreen(p1);
                  PopupMenu1->Popup(p2.x, p2.y);
                  return;
                }
                i++;
              }
            }
            ii++;
          }
          }////
        }

        }//
}
//---------------------------------------------------------------------------

// i.14. обработка для кнопки BMP - создание файла shema.bmp

void __fastcall Tform_dalvjaz::b_bmpClick(TObject *Sender)
{


          Graphics::TBitmap *Bitmap;
          struct  tm  *gmt;
          char stro[100], st[100];
          int l;
          TRect   MyRect, MyOther;

          /*
          gbs_fl_bmp = 1;

          zagr_gbs_old_smxy = 1;     zagruzka_gbs(mal[MDItek].tek_alg);
          zagr_gbs_old_smxy = 0;
          */

          dalvjaz_fl_bmp = 1;
          wywod_shemy(Sender);      dalvjaz_fl_bmp = 0;

          MyRect = Rect(0,0,PaintBox1->Width-2,PaintBox1->Height-2);
```

```
            Bitmap = new Graphics::TBitmap;
            Bitmap->Width = PaintBox1->Width;
            Bitmap->Height = PaintBox1->Height;
            Bitmap->Canvas->CopyRect(MyRect, PaintBox1->Canvas, MyRect);

            //strcpy(st, mal[MDItek].m_alg[mal[MDItek].tek_alg]->imq_zagol);

            strcpy(stro, "shema.");
            strcat(stro, "bmp");

            Bitmap->SaveToFile(stro);
            delete Bitmap;

            wywod_shemy(Sender);

            /*
            gbs_fl_bmp = 0;

            zagr_gbs_old_smxy = 1;     zagruzka_gbs(mal[MDItek].tek_alg);
            zagr_gbs_old_smxy = 0;
            */

}
//---------------------------------------------------------------------------

// i.15. обработка для горизонтальной полосы прокрутки BC

void __fastcall Tform_dalvjaz::ScrollBar2Change(TObject *Sender)
{
         int pos;

         pos = ScrollBar2->Position;

         dalvjaz_smx = -((pos*dalvjaz_xmax)/100);

         wywod_shemy(Sender);

}
//---------------------------------------------------------------------------

// i.16. обработка кнопки "к первой ветке"

void __fastcall Tform_dalvjaz::b_w_nClick(TObject *Sender)
{
         ScrollBar2->Position = 0;
         dalvjaz_tek_wetka = 0;

         form_ls->zagruzka_LSW(Sender);
}
//---------------------------------------------------------------------------

// i.17. обработка кнопки "к последней ветке"

void __fastcall Tform_dalvjaz::b_w_kClick(TObject *Sender)
{
         int tx;

         dalvjaz_tek_wetka = tek_alg.kol_wetok-2;

         if (dalvjaz_tek_wetka <= 0) {  b_w_nClick(Sender);  return;  }

         tx = tek_alg.m_x_wetok[dalvjaz_tek_wetka] - (dalvjaz_chw*5);
```

```
        ScrollBar2->Position = (tx*100)/dalvjaz_xmax;

        form_ls->zagruzka_LSW(Sender);
}
//---------------------------------------------------------------------------

// i.18. обработка кнопки "к предыдущей ветке"

void __fastcall Tform_dalvjaz::b_w_minClick(TObject *Sender)
{
        int tx;

        dalvjaz_tek_wetka--;

        if (dalvjaz_tek_wetka > (tek_alg.kol_wetok-2))
                                    dalvjaz_tek_wetka = tek_alg.kol_wetok-2;
        if (dalvjaz_tek_wetka <= 0)  {  b_w_nClick(Sender);  return;  }

        tx = tek_alg.m_x_wetok[dalvjaz_tek_wetka] - (dalvjaz_chw*5);

        ScrollBar2->Position = (tx*100)/dalvjaz_xmax;

        form_ls->zagruzka_LSW(Sender);
}
//---------------------------------------------------------------------------

// i.19. обработка кнопки "к следующей ветке"

void __fastcall Tform_dalvjaz::b_w_plClick(TObject *Sender)
{
        int tx;

        dalvjaz_tek_wetka++;

        if (dalvjaz_tek_wetka > (tek_alg.kol_wetok-2))
                                    dalvjaz_tek_wetka = tek_alg.kol_wetok-2;
        if (dalvjaz_tek_wetka <= 0){  b_w_nClick(Sender);  return;  }

        tx = tek_alg.m_x_wetok[dalvjaz_tek_wetka] - (dalvjaz_chw*5);

        ScrollBar2->Position = (tx*100)/dalvjaz_xmax;

        form_ls->zagruzka_LSW(Sender);
}
//---------------------------------------------------------------------------

// i.20. обработка движения мыши по ВС

void __fastcall Tform_dalvjaz::PaintBox1MouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
        int j, i, ii, k, x1, x2, y1, y2;


        wspl_inf_okno = NO;

        for (j=0; j < (tek_alg.kol_wetok-1); j++) {////

          ii = tek_alg.m_inew[j];

          i=0;
          while (i < tek_alg.m_el_wert[j]) {
```

```
            if (tek_alg.m_el[ii].n_wert == j) {

              if ((tek_alg.m_el[ii].used == IS) ||
                  (j == (tek_alg.kol_wetok-1))) {

                wsplyw_inf_okno(X, Y, j, ii);
                if (wspl_inf_okno == IS)   return;

                i++;
              }
            }
            ii++;
          }
        }////

        // вывод элементов дополнительных вертикалей веток

        for (j=0; j < tek_alg.kol_dwert; j++) {////


          if (tek_alg.m_el_wert[KOL_WETOK+j] != 0) {////

          ii = tek_alg.m_inew[KOL_WETOK+j];

          i=0;
          while (i < tek_alg.m_el_wert[KOL_WETOK+j]) {

            if (tek_alg.m_el[ii].n_wert == (KOL_WETOK+j)) {

              if (tek_alg.m_el[ii].used == IS) {

                wsplyw_inf_okno(X, Y, KOL_WETOK+j, ii);
                if (wspl_inf_okno == IS)  return;

                i++;
              }
            }
            ii++;
          }
          }////
        }////

        if (Panel4->Visible == true)   Panel4->Hide();
}
//---------------------------------------------------------------------------


// i.21. вывод схемы

void __fastcall Tform_dalvjaz::wywod_shemy(TObject *Sender)
{
        int i, j, x1, y1, x2, y2, y3, y4, len, ii, yw0, yw1,
            smxzl, smxzr, ipe;
        char stro[dl_stro], *u, st[50];


        PaintBox1->Canvas->Font->Name =AnsiString(cfg_imq_fonta_1);
        PaintBox1->Canvas->Font->Size =atoi(cfg_razmer_fonta_1);

        dalvjaz_chw = PaintBox1->Canvas->TextWidth("0");   // 1:7
        dalvjaz_chh = PaintBox1->Canvas->TextHeight("0");  // 1:15
        dalvjaz_chdy = dalvjaz_chh + 1;
```

```
if (dalvjaz_fl_bmp == 0)
  PaintBox1->Canvas->Brush->Color= RGB(200,200,200);
else
  PaintBox1->Canvas->Brush->Color= RGB(255,255,255);

PaintBox1->Canvas->FillRect
          (Rect(0,0,PaintBox1->Width,PaintBox1->Height));


// рисуем заголовок

if (dalvjaz_shema_schitana == NO)
              strcpy(stro,
              "___ Заголовок визуальной схемы ___");
else
              strcpy(stro, tek_alg.imq_zagol);

u = stro;
while (*u == ' ')  u++;

len = (strlen(stro)*dalvjaz_chw)+dalvjaz_chw*5;


if ((len/2) <= (dalvjaz_chw*7)) {

  smxzl = len/2;   smxzr = len/2;
}
else {
  smxzl = (dalvjaz_chw*7);
  smxzr = len - (dalvjaz_chw*7);
}

x1 = dalvjaz_smx + tek_alg.m_x_wetok[0]-smxzl;
x2 = dalvjaz_smx + tek_alg.m_x_wetok[0]+smxzr;
y1 = dalvjaz_smy + DALVJAZ_Y_ZAGOL;
y2 = dalvjaz_smy + DALVJAZ_Y_ZAGOL+((3*dalvjaz_chdy)/2);
yw0 = y2;

dalvjaz_line(x1,y1,x2,y1);
dalvjaz_line(x2,y1,x2,y2);
dalvjaz_line(x2,y2,x1,y2);
dalvjaz_line(x1,y2,x1,y1);

x1 += (dalvjaz_chw*2);
y1 = dalvjaz_smy + DALVJAZ_Y_ZAGOL + (dalvjaz_chdy / 3);

dalvjaz_str(x1,y1, u);


if ((tek_alg.kol_elem == 0) ||
    (dalvjaz_shema_schitana == NO)) {

  goto  m_end;  // если элементов нет или не проверены все
                // ветки схемы, то выход
}

podschet_max_koord(Sender);


// вертикальная линия от заголовка до верхней горизонтальной
// линии силуэта

y1 = yw0;
x1 = dalvjaz_smx + tek_alg.m_x_wetok[0];    x2 = x1;
```

```
y2 = y1 + (dalvjaz_chdy*2) + (dalvjaz_chdy/2);
y3 = y1 + dalvjaz_chdy;

dalvjaz_line(x1,y1,x2,y3);

yw0 = y2;  // начальная y для всех веток

// рисуем верхнюю горизонтальную линию силуэта

x1 = dalvjaz_smx + tek_alg.m_x_wetok[0] - (dalvjaz_chw*5);
x2 = x1;

for (i=0; i < tek_alg.kol_wetok-1; i++)
                      x2 += tek_alg.m_dx_wetok[i];
x2 += (dalvjaz_chw*5);

dalvjaz_line(x1,y3,x2,y3);

yw1 = y3;

// стрелка обратной петли силуэта

x2 = dalvjaz_smx + tek_alg.m_x_wetok[0];   x1 = x2 - (dalvjaz_chw*2);
y4 = y3 - (dalvjaz_chdy / 4);
dalvjaz_line(x1,y4,x2,y3);
y4 = y3 + (dalvjaz_chdy / 4);
dalvjaz_line(x1,y4,x2,y3);

/**/

// рисуем все вертикали

for (j=0; j < tek_alg.kol_wetok; j++) {////

  //tek_alg.m_wert[j].y1 = yw0;

  x1 = dalvjaz_smx + tek_alg.m_wert[j].x;
  dalvjaz_line(x1,yw0,x1,yw1);

  ii = tek_alg.m_inew[j];

  i=0;
  while (i < tek_alg.m_el_wert[j]) {

    if (tek_alg.m_el[ii].n_wert == j) {

      if ((tek_alg.m_el[ii].used == IS) ||
          (j == (tek_alg.kol_wetok-1))) {

        wywod_elem(Sender, j, ii, i);   i++;  }
    }

    ii++;
  }
}////

// вывод элементов дополнительных вертикалей веток

for (j=0; j < tek_alg.kol_dwert; j++) {////

  x1 = tek_alg.m_wert[tek_alg.m_wert[KOL_WETOK+j].parent_w].x +
                         dalvjaz_smx + (3*dalvjaz_chw);
  x2 = dalvjaz_smx + tek_alg.m_wert[KOL_WETOK+j].x;
  y1 = dalvjaz_smy + tek_alg.m_wert[KOL_WETOK+j].y1;
```

```
  if (tek_alg.m_wert[KOL_WETOK+j].type == T_WERT_DOCHX) {

    dalvjaz_line(x1,y1,x2,y1);

    ipe = tek_alg.m_wert[KOL_WETOK+j].parent_endw;
                                    //tek_alg.m_el[ipe].n_wert
    x1 = dalvjaz_smx + tek_alg.m_wert[ipe].x;
    x2 = dalvjaz_smx + tek_alg.m_wert[KOL_WETOK+j].x;
    y1 = dalvjaz_smy + tek_alg.m_wert[KOL_WETOK+j].y1 +
                                   tek_alg.m_wert[KOL_WETOK+j].tyel;
    dalvjaz_line(x1,y1,x2,y1);

    if (tek_alg.m_el_wert[KOL_WETOK+j] != 0)
    if (tek_alg.m_wert[KOL_WETOK+j].type_1_el != t_EB) {

      y1 = dalvjaz_smy + tek_alg.m_wert[KOL_WETOK+j].y1;
      y2 = y1 + tek_alg.m_el[ tek_alg.m_inew[KOL_WETOK+j] ].y1;

      dalvjaz_line(x2,y1,x2,y2);
    }
  }
  if (tek_alg.m_wert[KOL_WETOK+j].type == T_WERT_CIKLA) {

    x1 = x1 - (3*dalvjaz_chw);
    dalvjaz_line(x1,y1,x2,y1);

    // стрелка цикла

    dalvjaz_line(x1,y1,x1+dalvjaz_chw,y1-(dalvjaz_chw/2));
    y4 = y3 + (dalvjaz_chdy / 4);
    dalvjaz_line(x1,y1,x1+dalvjaz_chw,y1+(dalvjaz_chw/2));


    y2 = y1 + tek_alg.m_wert[KOL_WETOK+j].tyel;
    x1 = x1 + (3*dalvjaz_chw);
    dalvjaz_line(x1,y2,x2,y2);
  }

  if (tek_alg.m_el_wert[KOL_WETOK+j] == 0) {

    y1 = dalvjaz_smy + tek_alg.m_wert[KOL_WETOK+j].y1;
    y2 = y1 + tek_alg.m_wert[KOL_WETOK+j].tyel;
    dalvjaz_line(x2,y1,x2,y2);
  }
  else {////

  ii = tek_alg.m_inew[KOL_WETOK+j];

  i=0;
  while (i < tek_alg.m_el_wert[KOL_WETOK+j]) {

    if (tek_alg.m_el[ii].n_wert == (KOL_WETOK+j)) {

      if (tek_alg.m_el[ii].used == IS) {

        wywod_elem(Sender, KOL_WETOK+j, ii, i);   i++;  }
    }

    ii++;
  }
  }////
}////
```

```
        // нижняя горизонтальная линия силуэта

        x1 = dalvjaz_smx + tek_alg.m_x_wetok[0] - (dalvjaz_chw*5);

        for (i=0; i < tek_alg.kol_wetok-1; i++)
                              x2 = dalvjaz_smx + tek_alg.m_x_wetok[i];

        dalvjaz_line(x1, dalvjaz_smy+dalvjaz_ymax,
                     x2, dalvjaz_smy+dalvjaz_ymax);

        // вертикальная линия обратной петли силуэта

        dalvjaz_line(x1, yw0 -((dalvjaz_chdy*3)/2),
                     x1, dalvjaz_smy+dalvjaz_ymax);

        m_end:
}

//---------------------------------------------------------------------------

// i.22. вывод элемента схемы

void __fastcall Tform_dalvjaz::wywod_elem(TObject *Sender, int w, int ie, int n)
{
        int x1, x2, x3, x4, y1, y2, y3, yww, len, dxb, dyb, smxw, ii;
        char st[50], st2[50];


        TRect   RectP, RectB = Rect(0,0,100,100);

        dxb = RAZM_X_BMP;
        dyb = RAZM_Y_BMP;
        RectB.Right = dxb;
        RectB.Bottom = dyb;


        y1 = dalvjaz_smy + tek_alg.m_wert[w].y1 + tek_alg.m_el[ie].y1;
        y2 = dalvjaz_smy + tek_alg.m_wert[w].y1 + tek_alg.m_el[ie].y2;

        x1 = dalvjaz_smx + tek_alg.m_wert[w].x + tek_alg.m_el[ie].x1;
        x2 = dalvjaz_smx + tek_alg.m_wert[w].x + tek_alg.m_el[ie].x2;
        yww = y2;

        if ((tek_alg.m_el[ie].type == t_E) ||
            (tek_alg.m_el[ie].type == t_EB))  goto  m_liniq;


        if ((tek_alg.m_el[ie].type == t_A)   ||
            (tek_alg.m_el[ie].type == t_RET) ||
            (tek_alg.m_el[ie].type == t_CB)) {

          dalvjaz_line(x1,y1,x2,y1);
          dalvjaz_line(x2,y1,x2,y2);
          dalvjaz_line(x1,y1,x1,y2);
          dalvjaz_line(x1,y2,x2,y2);

        } else if (tek_alg.m_el[ie].type == t_HD) {

          y3 = y2 - ((3*dalvjaz_chdy)/8);
          dalvjaz_line(x1,y1,x2,y1);
          dalvjaz_line(x2,y1,x2,y3);
          dalvjaz_line(x1,y1,x1,y3);
```

```
  dalvjaz_line(x1,y3,dalvjaz_smx + tek_alg.m_wert[w].x,y2);
  dalvjaz_line(x2,y3,dalvjaz_smx + tek_alg.m_wert[w].x,y2);

} else if (tek_alg.m_el[ie].type == t_AD) {

  y3 = y1 + ((3*dalvjaz_chdy)/8);

  dalvjaz_line(x1,y3,dalvjaz_smx + tek_alg.m_wert[w].x,y1);
  dalvjaz_line(x2,y3,dalvjaz_smx + tek_alg.m_wert[w].x,y1);
  dalvjaz_line(x1,y3,x1,y2);
  dalvjaz_line(x2,y3,x2,y2);
  dalvjaz_line(x1,y2,x2,y2);

} else if ((tek_alg.m_el[ie].type == t_IF) ||
           (tek_alg.m_el[ie].type == t_EI) ||
           (tek_alg.m_el[ie].type == t_CE)) {

  x3 = x1 + dalvjaz_chw;    x4 = x2 - dalvjaz_chw;
  y3 = (y1+y2)/2;
  dalvjaz_line(x1,y3,x3,y1);
  dalvjaz_line(x1,y3,x3,y2);
  dalvjaz_line(x3,y1,x4,y1);
  dalvjaz_line(x3,y2,x4,y2);
  dalvjaz_line(x4,y1,x2,y3);
  dalvjaz_line(x4,y2,x2,y3);
}


if (tek_alg.m_el[ie].n_bmp == -1) {////

// текст элемента

x3 = x1 + dalvjaz_chw;
y3 = y1 + dalvjaz_chdy / 3;

if (tek_alg.m_el[ie].type == t_HD)  y3 -= dalvjaz_chdy / 6;
if (tek_alg.m_el[ie].type == t_AD)  y3 += dalvjaz_chdy / 6;

if (tek_alg.m_el[ie].name[0] != 0)

  dalvjaz_str(x3,y3, tek_alg.m_el[ie].name);

else {

  if (tek_alg.m_el[ie].type == t_HD) {
    strcpy(st, " B");

    itoa(tek_alg.m_el[ie].wetka, st2, 10);
    strcat(st,st2);
  }
  if (tek_alg.m_el[ie].type == t_AD) {
    strcpy(st, " B");
    if (tek_alg.m_el[ie].wetka2 == 0)  strcpy(st, "ВЫХ.");
    else {
      if (tek_alg.m_el[ie].wetka2 == -1)  strcpy(st2, "???");
      else
        itoa(tek_alg.m_el[ie].wetka2, st2, 10);
      strcat(st,st2);
    }
  }
  if (tek_alg.m_el[ie].type == t_IF)  strcpy(st, " IF ");
  if (tek_alg.m_el[ie].type == t_EI)  strcpy(st, " EI ");
  if (tek_alg.m_el[ie].type == t_CE)  strcpy(st, " CE ");
  if (tek_alg.m_el[ie].type == t_CB)  strcpy(st, " CB ");
```

```
  if (tek_alg.m_el[ie].type == t_A)   strcpy(st, " A  ");
  if (tek_alg.m_el[ie].type == t_RET) strcpy(st, "КОН.");

  dalvjaz_str(x3,y3, st);
}

}
else {//// вывод пиктограммы

if (((tek_alg.m_el[ie].type == t_A)  ||
     (tek_alg.m_el[ie].type == t_CE) ||
     (tek_alg.m_el[ie].type == t_IF) ||
     (tek_alg.m_el[ie].type == t_EI)) &&
    (strcmp(tek_alg.m_el[ie].name, "КОН.") != 0)) {
  RectP.Left = x1+SM_X_PIKTOGRAMMY;
  RectP.Top  = y1+SM_Y_PIKTOGRAMMY;
  RectP.Right = x1+SM_X_PIKTOGRAMMY+dxb;
  RectP.Bottom = y1+SM_Y_PIKTOGRAMMY+dyb;

  PaintBox1->Canvas->CopyRect(RectP,
    m_Bitmap[tek_alg.m_el[ie].n_bmp]->Canvas, RectB);
}

}////


// индекс элемента

if (tek_alg.m_el[ie].ind != 0) {

  itoa(tek_alg.m_el[ie].ind, st, 10);     len = strlen(st);
  x3 = x1 + ((2-len)*dalvjaz_chw);
  y3 = y1 - dalvjaz_chdy+1;

  dalvjaz_str(x3,y3, st);

  // подчеркивание индекса, если у элемента есть ссылка
  //if (tek_alg.m_el[ie].wetka == 1)
  //  dalvjaz_line(x3, y3+dalvjaz_chh-2,
  //             x3+(len*dalvjaz_chw), y3+dalvjaz_chh-2);
}

// ДА/НЕТ

if ((tek_alg.m_el[ie].type == t_CE) ||
    (tek_alg.m_el[ie].type == t_IF) ||
    (tek_alg.m_el[ie].type == t_EI)) {

  if (tek_alg.m_el[ie].fl_usl == IS) {
    strcpy(st, "да");   strcpy(st2, "нет");  }
  else {
    strcpy(st2, "да");   strcpy(st, "нет");  }

  dalvjaz_str(x1+(6*dalvjaz_chw)+2,y1-3, st2);
  dalvjaz_str(x1+(4*dalvjaz_chw)-4,y2+1, st);



}


// рисуем линию к следующему элементу

m_liniq:
```

```
          if ((w == (tek_alg.kol_wetok-1)) && (n == (tek_alg.m_el_wert[w]-1)))

            return;  // для последнего эл. метки ВЫХОД не нужно

          // проводим линию до следующего эл-та или, если эл. последний на
          // вертикали, то до  tyel  вертикали

          y1 = y2;
          x1 = dalvjaz_smx + tek_alg.m_wert[w].x;    x2 = x1;

          //y2 = y1 + dalvjaz_chdy;
          //if (tek_alg.m_el[ie].type == t_IF)  y2 += dalvjaz_chdy / 2;

          if (n == (tek_alg.m_el_wert[w]-1)) {

            if (tek_alg.m_wert[w].type == T_WERT_GLAWNAQ)
                              y2 = dalvjaz_smy + dalvjaz_ymax;
            else
                              y2 = dalvjaz_smy + tek_alg.m_wert[w].y1 +
                                                 tek_alg.m_wert[w].tyel;
          }
          else {
            ii = ie + 1;

            while (ii < KOL_ELEM) {

              if (((tek_alg.m_el[ii].used == IS) || (ii >= MAX_KOL_ELEM)) &&
                  (tek_alg.m_el[ii].n_wert == tek_alg.m_el[ie].n_wert)) {

                y2 = dalvjaz_smy + tek_alg.m_wert[w].y1 + tek_alg.m_el[ii].y1;
                break;
              }
              ii++;
            }
          }

          dalvjaz_line(x1,y1,x2,y2);
}

//---------------------------------------------------------------------------

// i.23. подсчитываем размер схемы

void __fastcall Tform_dalvjaz::podschet_max_koord(TObject *Sender)
{
          int i, ii, j, jj, k, y, tw, tdx, tdx2;


          tek_alg.kol_dwert = 0;
          i= KOL_WETOK;
          while (i < MAX_KOL_WERT) {
            if (tek_alg.m_wert[i].used == IS)   tek_alg.kol_dwert++;
            i++;
          }

          // подсчет  m_dx_wetok[tw], tek_alg.m_wert[w].x  в зависимости
          // от наличия дополнительных вертикалей

          // m_dx_wetok

          // для всех веток найдем максимальное смещение от главной
          // вертикали
```

```
for (j=0; j < (tek_alg.kol_wetok-1); j++) {

  ii= KOL_WETOK;   i= 0;   k = 0;   tdx = 0;
  while (i < tek_alg.kol_dwert) {

    if ((tek_alg.m_wert[ii].used == IS) &&
        (tek_alg.m_wert[ii].wetka == (j+1)))  {

      tdx2 = (tek_alg.m_wert[ii].n_sprawa+1)*dalvjaz_chw*8;
      if (tdx2 > tdx)   tdx = tdx2;
      k++;
    }
    ii++;   i++;
  }
  if (k > 0)    tek_alg.m_dx_wetok[j] = tdx;
  else          tek_alg.m_dx_wetok[j] = dalvjaz_chw*8;
}

// m_x_wetok

for (i=0; i < tek_alg.kol_wetok; i++) {
  tek_alg.m_wert[i].x = OTSTUP_WERT+(dalvjaz_chw*5);
  for (ii=0; ii < i; ii++)
    tek_alg.m_wert[i].x += tek_alg.m_dx_wetok[ii];

  tek_alg.m_x_wetok[i] = tek_alg.m_wert[i].x;
}

if (tek_alg.kol_dwert > 0) {//

  // смещения дочерних вертикалей

  i= KOL_WETOK;  j=0;
  while (j < tek_alg.kol_dwert) {

    if (tek_alg.m_wert[i].used == IS) {

      tw = tek_alg.m_wert[i].wetka-1;
      tek_alg.m_wert[i].x = tek_alg.m_wert[tw].x +
                    (tek_alg.m_wert[i].n_sprawa * dalvjaz_chw*8);
    }

    i++;   j++;
  }

}


dalvjaz_xmax = OTSTUP_WERT;
for (i=0; i < tek_alg.kol_wetok; i++)
                              dalvjaz_xmax += tek_alg.m_dx_wetok[i];
dalvjaz_xmax += dalvjaz_chw*3;


dalvjaz_ymax = 0;

for (j=0; j < tek_alg.kol_wetok; j++) {

  y = tek_alg.m_wert[j].y1 + tek_alg.m_wert[j].tyel;

  if (y > dalvjaz_ymax)   dalvjaz_ymax = y;

}
```

```
for (j=0; j < (tek_alg.kol_wetok-1); j++)
  tek_alg.m_wert[j].tyel = dalvjaz_ymax - tek_alg.m_wert[j].y1;


// для всех главных вертикалей, если t_E последний, то
// опускаем t_E на уровень нижней горизонтали схемы и то же с
// tyel всех вертикалей, для которых этот t_E закрывающий

for (j=0; j < (tek_alg.kol_wetok-1); j++) {

  ii = tek_alg.m_inew[j];  i = 0;   k = 0;
  while ((ii < MAX_KOL_ELEM) && (i < tek_alg.m_el_wert[j])) {

    if ((tek_alg.m_el[ii].used == IS) &&
        (tek_alg.m_el[ii].n_wert == j)) {

       i++;       mie_wetki[k] = ii;   k++;
    }
    ii++;
  }

  for (i=(k-1); i > 0; i--) {

    if (tek_alg.m_el[mie_wetki[i]].type == t_E) {

      tek_alg.m_el[mie_wetki[i]].end_w = IS;
      tek_alg.m_el[mie_wetki[i]].y1 = tek_alg.m_wert[j].tyel;
      tek_alg.m_el[mie_wetki[i]].y2 = tek_alg.m_wert[j].tyel;

      // для текущего эл. ищем все дочерние вертикали, для
      // которых он является закрывающим, и корректируем у
      // них tyel

      ii = KOL_WETOK;   jj = 0;
      while ((ii < MAX_KOL_WERT) && (jj < tek_alg.kol_dwert)) {

        if (tek_alg.m_wert[ii].used == IS) {

          if (tek_alg.m_wert[ii].parent_end == mie_wetki[i]) {

            tek_alg.m_wert[ii].tyel =
                              dalvjaz_ymax - tek_alg.m_wert[ii].y1;
          }
          jj++;
        }
        ii++;
      }
    }
    else  break;
  }
}

// для всех вертикалей, у которых tyel упирается в нижнюю
// горизонталь и t_AD является последним элементом, лежащим
// выше t_E на нижней горизонтали, опускаем t_AD к нижней
// горизонтали

jj = tek_alg.kol_wetok-1 + tek_alg.kol_dwert;

ii = 0;   k=0;
while ((ii < MAX_KOL_WERT) && (k < jj)) {

  if (tek_alg.m_wert[ii].used == IS) {
```

```
            if (tek_alg.m_wert[ii].tyel ==
                       (dalvjaz_ymax - tek_alg.m_wert[ii].y1)) {

              j = tek_alg.m_inew[ii];  i=0;
              while ((j < MAX_KOL_ELEM) && (i < tek_alg.m_el_wert[ii])) {

                if ((tek_alg.m_el[j].used == IS) &&
                    (tek_alg.m_el[j].n_wert == ii)) {

                  mie_wetki[i] = j;
                  i++;
                }
                j++;
              }

              for (tw=(i-1); tw > 0; tw--) {

                if ((tek_alg.m_el[mie_wetki[tw]].type != t_E) &&
                    (tek_alg.m_el[mie_wetki[tw]].type != t_AD))  break;
                else
                if (tek_alg.m_el[mie_wetki[tw]].type == t_AD) {

                  tek_alg.m_el[mie_wetki[tw]].y2 =
                         tek_alg.m_wert[ii].tyel - dalvjaz_chdy;

                  tek_alg.m_el[mie_wetki[tw]].y1 =
                    tek_alg.m_el[mie_wetki[tw]].y2 -((3*dalvjaz_chdy)/2);
                  break;
                }
              }

            }
            k++;
          }

          ii++;
        }
}

//---------------------------------------------------------------------------


// i.24. обработка закрытия формы BC

void __fastcall Tform_dalvjaz::FormClose(TObject *Sender,
      TCloseAction &Action)
{
        form_ls->Close();
}
//---------------------------------------------------------------------------

// i.25. удалить заданное число начальных пробелов в строке и,
//       если требуется, оставить между словами строки только 1 пробел

void  udal_probely(char *str, int nach, uchar fl_1_prob)
{
        int l, i, j, kp;
        char stro[dl_stro];

        if (nach != 0) {

          l = strlen(str);

          for (i=nach; i <= l; i++)   str[i-nach] = str[i];
```

```
        }

        if (fl_1_prob == IS) {

          strcpy(stro, str);    l = strlen(stro);   j = 0;   kp = 0;

          for (i=0; i < l; i++) {

            if (stro[i] != ' ') {  str[j] = stro[i];   j++;   kp = 0;   }
            else  {
              if (kp < 1) {  str[j] = stro[i];   j++;   }
              kp++;
            }
          }

          str[j] = 0;
        }
}


// i.26. формирование элемента при чтении схемы из файла

void  dalvjaz_form_el(int type, int te, int tw, int *ti, char *_up)
{
        int ii, l, i;
        char *_u, *_u2;


        tek_alg.m_el[te].type    = type;
        tek_alg.m_el[te].wetka   = tw+1;
        tek_alg.m_el[te].used    = IS;

        if ((tek_alg.m_el[te].type != t_E) &&
            (tek_alg.m_el[te].type != t_EB))  {
                              tek_alg.m_el[te].ind = *ti;  (*ti)++;  }
        else
          tek_alg.m_el[te].ind = 0;

        tek_alg.m_el[te].n_wert  = tw;
        tek_alg.m_el[te].n_dwert = -1;
        tek_alg.m_el[te].kol_s_d = 0;
        tek_alg.m_el[te].kol_s_k = 0;
        tek_alg.m_el[te].n_bmp   = -1;
        tek_alg.m_el[te].end_w   = NO;

        tek_alg.m_el_wetki[tw]++;

        if (_up == NULL)  return;

        _u = _up;   while (((*_u) != ' ') && ((*_u) != 0)) _u++;

        if ((*_u) == 0)  return;

        while (((*_u) == ' ') && ((*_u) != 0)) _u++;

        if ((*_u) == 0)  return;

        _u2 = strstr(_u, ".bmp");
        if (_u2 != NULL) {// bmp

          for (i=0; i < RAZM_M_BITMAP; i++) {

            if (strcmp(m_imen_bmp[i], _u) == 0) {
```

```
              tek_alg.m_el[te].n_bmp = i;   break;
            }
          }


        }
        else {// сокращение

          l = strlen(_u);      if (l > 4)  l = 4;

          for (i=0; i < l; i++) {
            tek_alg.m_el[te].name[i] = *_u;   _u++;  }

          tek_alg.m_el[te].name[i] = 0;
        }
}


// i.27.

// i.28. закончить формирование элемента

void  zakonchitx_wwod_el(int te)
{
        if (te != -1) {// заканчиваем ввод элемента
          // если последняя строка массива действий пуста или в ней
          // только пробелы, то удаляем ее

          if ((tek_alg.m_el[te].type == t_E) ||
              (tek_alg.m_el[te].type == t_EB))   return;

          while (tek_alg.m_el[te].kol_s_d > 0) {

            if (str_probelow(tek_alg.m_el[te].m_d[
                  tek_alg.m_el[te].kol_s_d-1].m) != IS)
                                                  break;
            tek_alg.m_el[te].kol_s_d--;
          }
        }
}


// i.29. чтение ВС из файла

void __fastcall Tform_dalvjaz::b_chtenieClick(TObject *Sender)
{
        FILE *fl;
        int i, n, te, tw, ii,
        nprob,  // число удаляемых пробелов в начале строки
        ti = 1; // текущий индекс на схеме
        char stro[dl_stro], *_u, fname[80],
             *_upne, // указатель на признак начала элемента
             stro2[dl_stro], st[50];
        uchar fl_cht_zgl;

        dalvjaz_shema_schitana = NO;

        dalvjaz_ymax = 0;


        te = -1;  // текущего элемента нет

        fl_cht_zgl = NO;
```

```
dalvjaz_smx = 0;   ScrollBar2->Position = 0;
dalvjaz_smy = 0;   ScrollBar1->Position = 0;

// ставим нужный фонт
form_dalvjaz->PaintBox1->Canvas->Font->Name
                                  =AnsiString(cfg_imq_fonta_1);
form_dalvjaz->PaintBox1->Canvas->Font->Size=atoi(cfg_razmer_fonta_1);

dalvjaz_chw = form_dalvjaz->PaintBox1->Canvas->TextWidth("0");  // 1:7
dalvjaz_chh = form_dalvjaz->PaintBox1->Canvas->TextHeight("0"); // 1:15
dalvjaz_chdy = dalvjaz_chh + 1;

//======== очистка массивов данных схемы ===============

strcpy(tek_alg.imq_zagol, "Заголовок визуальной схемы");

for (i=0; i < KOL_ELEM; i++)  tek_alg.m_el[i].used = NO;
// 2 элемента (MAX_KOL_ELEM, MAX_KOL_ELEM+1) оставляем для ветки
// ВЫХОД, которая есть всегда
// у эл-тов ветки ВЫХОД флаг used всегда сброшен

// tek_alg.kol_wetok - будет задаваться с учетом ветки ВЫХОД
// tek_alg.kol_elem  и  tek_alg.kol_dwert  будут задаваться
// без учета ветки ВЫХОД

tek_alg.kol_wetok = 0;
tek_alg.kol_elem = 0;
tek_alg.kol_dwert = 0;

// для ветки 1 задаем всегда
tek_alg.m_x_wetok[0] = OTSTUP_WERT + (dalvjaz_chw * 5);
tek_alg.m_dx_wetok[0] = dalvjaz_chw * 8;

for (i=0; i < KOL_WERT; i++)   tek_alg.m_wert[i].used = NO;

// пока у веток нет дополнительных вертикалей :
for (i=0; i < KOL_WETOK; i++) {

  tek_alg.m_dw_wetki[i] = 0;
  tek_alg.m_inwwet[i]   = -1;
}

//=======================================================

strcpy(fname, "shema.prg");
if (fl_otkata_shemy == IS)   strcpy(fname, "shema.otk");
else {

  if (!FileExists(AnsiString("shema.prg"))) {
    // создаем файл по умолчанию

    fl=fopen("shema.prg", "wt");

    fprintf(fl, "\n// i.1. заголовок схемы\n\n");
    fprintf(fl, "HD  1       //.1\n");
    fprintf(fl, "  AD  UNDEF //.2\n");
    fprintf(fl, "HD  EXIT    //.3\n");
    fprintf(fl, "RETURN      //.4\n");
    fclose(fl);
  }
}

fl=fopen(fname, "rt");
if (fl == NULL)  return;
```

```
while (fgets(stro, 199, fl) != NULL)  {
  i = strlen(stro);
  stro[i-1] = 0;   i--;

  if (fl_cht_zgl == IS) {////

    _upne = strstr(stro, "//.");
    if (_upne != NULL) {// начало элемента схемы

      zakonchitx_wwod_el(te);

      nprob = 0;   while (stro[nprob] == ' ')   nprob++;

      udal_probely(stro, nprob, IS);
      _upne = strstr(stro, "//.");  // после удаления пробелов

      if (tek_alg.kol_elem == MAX_KOL_ELEM) {
        form_ls->osh_sozd_shemy = SHEMA_m_elem_full;
        form_ls->obrabotka_oshibok(Sender);
        tek_alg.kol_elem = 0;
        return;
      }

      tek_alg.kol_elem++;
      te = tek_alg.kol_elem-1;   tw = tek_alg.kol_wetok-1;

      if (tek_alg.kol_wetok == 0) {

        _u = strstr(stro, "HD");

        if (_u == NULL) {

          form_ls->osh_sozd_shemy = FAJL_net_zagol_w1;
          form_ls->obrabotka_oshibok(Sender);
          tek_alg.kol_elem = 0;
          return;
        }
        else  goto  m_hd;
      }

      _u = strstr(stro, "ELSE IF");
      if (_u != NULL) {
        if (stro[8] == '+')  tek_alg.m_el[te].fl_usl = IS;
        else                 tek_alg.m_el[te].fl_usl = NO;
        dalvjaz_form_el(t_EI, te, tw, &ti, _upne);
      }
      else {
        _u = strstr(stro, "IF");
        if (_u != NULL) {
          if (stro[3] == '+')  tek_alg.m_el[te].fl_usl = IS;
          else                 tek_alg.m_el[te].fl_usl = NO;
          dalvjaz_form_el(t_IF, te, tw, &ti, _upne);
        }
        else {
          _u = strstr(stro, "CB");
          if (_u != NULL) {
            dalvjaz_form_el(t_CB, te, tw, &ti, _upne);
          }
          else {
            _u = strstr(stro, "CE");
            if (_u != NULL) {
              if (stro[3] == '+')  tek_alg.m_el[te].fl_usl = IS;
              else                 tek_alg.m_el[te].fl_usl = NO;
              dalvjaz_form_el(t_CE, te, tw, &ti, _upne);
```

```
}
else {
  _u = strstr(stro, "HD");
  if (_u != NULL) {

    m_hd:

    if (tek_alg.kol_wetok == KOL_WETOK) {
      form_ls->osh_sozd_shemy = SHEMA_m_wetok_full;
      form_ls->obrabotka_oshibok(Sender);
      tek_alg.kol_elem = 0;
      return;
    }

    if (strstr(stro, "EXIT") != NULL) {

      // ветка ВЫХОД - конец чтения файла

      tek_alg.kol_elem--;
      break;
    }


    tek_alg.kol_wetok++;   tw = tek_alg.kol_wetok-1;

    tek_alg.m_wert[tw].type    = T_WERT_GLAWNAQ;
    tek_alg.m_el_wetki[tw]     = 0;
    tek_alg.m_inewet[tw]       = te;
    tek_alg.m_dw_wetki[tw]     = 0;
    tek_alg.m_inwwet[tw]       = -1;

    tek_alg.m_dx_wetok[tw] = dalvjaz_chw * 8;
      // если нет дочерних вертикалей

    tek_alg.m_x_wetok[tw] = OTSTUP_WERT+(dalvjaz_chw*5);
    for (ii=0; ii < tw; ii++)
      tek_alg.m_x_wetok[tw] += tek_alg.m_dx_wetok[ii];


    strcpy(tek_alg.m_el[te].name, "");

    dalvjaz_form_el(t_HD, te, tw, &ti, _upne);
  }
  else {
    _u = strstr(stro, "AD");
    if (_u != NULL) {

      strcpy(tek_alg.m_el[te].name, "");

      if ((*(_u+3)=='/') || (*(_u+3)=='U'))    {
         // т.к. несколько пробелов переводим в один
         // то если у AD нет адреса, то через пробел
         // после него будет '/'
        tek_alg.m_el[te].wetka2 = -1;
      }
      else if (*(_u+3)=='E') {

        tek_alg.m_el[te].wetka2 = 0;
      }
      else {
        st[0] = *(_u+3);   st[1] = *(_u+4);   st[3] = 0;
        tek_alg.m_el[te].wetka2 = atoi(st);
      }
```

```
                dalvjaz_form_el(t_AD, te, tw, &ti, _upne);
              }
              else {// обработка действия

                strcpy(tek_alg.m_el[te].name, "");

                dalvjaz_form_el(t_A, te, tw, &ti, _upne);

                _u = _upne;  _u--;
                while ((*_u) == ' ')  _u--;
                _u++;
                *_u = 0;

                if (str_probelow(stro) == NO) {
                  strcpy(tek_alg.m_el[te].m_d[0].m, stro);
                  tek_alg.m_el[te].kol_s_d = 1;
                }
              }
            }
          }
        }
      }
    }
    else {
      _u = strstr(stro, "ELSE");
      if (_u != NULL) {
        tek_alg.kol_elem++;
        te = tek_alg.kol_elem-1;   tw = tek_alg.kol_wetok-1;

        dalvjaz_form_el(t_EB, te, tw, &ti, NULL);
      }
      else {
        _u = strstr(stro, "END");
        if (_u != NULL) {
          tek_alg.kol_elem++;
          te = tek_alg.kol_elem-1;   tw = tek_alg.kol_wetok-1;

          dalvjaz_form_el(t_E, te, tw, &ti, NULL);
        }
        else {
          _u = strstr(stro, "// i.");
          if (_u != NULL) {// начало заголовка схемы

            break;  // уже был заголовок
          }
          else {// продолжение ввода элемента

            if (tek_alg.m_el[te].kol_s_d < KOL_STR_DEJSTW) {
              udal_probely(stro, nprob, NO);
              strcpy(tek_alg.m_el[te].m_d[tek_alg.m_el[te].kol_s_d].m,
                   stro);
              tek_alg.m_el[te].kol_s_d++;
            }
          }
        }
      }
    }
  }
}////
else {
  _u = strstr(stro, "// i.");
  if (_u != NULL) {// начало заголовка схемы

    fl_cht_zgl = IS;   _u += 3;
```

```
      strcpy(tek_alg.imq_zagol, _u);
    }
    else {

      if (i > 0) {
        n = 0;   while (stro[n] == ' ')   n++;

        if (n < i) {

          form_ls->osh_sozd_shemy = FAJL_net_zagol_shemy;
          form_ls->obrabotka_oshibok(Sender);
          tek_alg.kol_elem = 0;
          return;
        }
      }
    }
  }
}

fclose(fl);

if (fl_cht_zgl == NO) {
  form_ls->osh_sozd_shemy = FAJL_net_zagol_shemy;
  form_ls->obrabotka_oshibok(Sender);
  tek_alg.kol_elem = 0;
  return;
}

zakonchitx_wwod_el(te);

tek_alg.kol_wetok++;  // учитываем ветку ВЫХОД

// элементы ветки ВЫХОД

tw = tek_alg.kol_wetok - 1;

tek_alg.m_el_wetki[tw] = 2;
tek_alg.m_inewet[tw] = MAX_KOL_ELEM;
tek_alg.m_inwwet[tw] = -1;
tek_alg.m_dx_wetok[tw] = dalvjaz_chw * 8;

tek_alg.m_wert[tw].type = T_WERT_GLAWNAQ;
tek_alg.m_wert[tw].y1   = DALVJAZ_Y_ZAGOL + ((5*dalvjaz_chdy)/2);
tek_alg.m_wert[tw].tyel = dalvjaz_chdy + (dalvjaz_chdy/2);

tek_alg.m_el_wert[tw]      = 2;
tek_alg.m_inew[tw]         = MAX_KOL_ELEM;

// элементы метки ВЫХОД

te = MAX_KOL_ELEM;

tek_alg.m_el[te].ind  = 0;
tek_alg.m_el[te].wetka = tw+1;
tek_alg.m_el[te].type = t_HD;
strcpy(tek_alg.m_el[te].name, "ВЫХ.");
tek_alg.m_el[te].n_wert = tw;
tek_alg.m_el[te].n_bmp = -1;

tek_alg.m_el[te].x1 = -3 * dalvjaz_chw;
tek_alg.m_el[te].x2 = 3 * dalvjaz_chw;
tek_alg.m_el[te].y1 = tek_alg.m_wert[tw].tyel;
tek_alg.m_el[te].y2 = tek_alg.m_el[te].y1+((3*dalvjaz_chdy)/2);
```

```
          tek_alg.m_wert[tw].tyel = tek_alg.m_el[te].y2 + dalvjaz_chdy;
          te++;
          tek_alg.m_el[te].ind  = 0;
          tek_alg.m_el[te].wetka = tw+1;
          tek_alg.m_el[te].type = t_A;
          strcpy(tek_alg.m_el[te].name, "КОН.");
          tek_alg.m_el[te].n_wert = tw;
          tek_alg.m_el[te].n_bmp = -1;

          tek_alg.m_el[te].x1 = -3 * dalvjaz_chw;
          tek_alg.m_el[te].x2 = 3 * dalvjaz_chw;
          tek_alg.m_el[te].y1 = tek_alg.m_wert[tw].tyel;
          tek_alg.m_el[te].y2 = tek_alg.m_el[te].y1+((3*dalvjaz_chdy)/2);
          tek_alg.m_wert[tw].tyel = tek_alg.m_el[te].y2 + dalvjaz_chdy;

          //---------------------------------------

          // вызываем проверку ЛСВ для всех веток

          for (i=0; i < (tek_alg.kol_wetok-1); i++) {

            dalvjaz_tek_wetka = i;
            form_ls->zagruzka_LSW(Sender);
            form_ls->b_sozdatx_shemuClick(Sender);
            if (form_ls->osh_sozd_shemy != SHEMA_OK) {

              return;
            }
          }

          read_comment(fname);

          dalvjaz_shema_schitana = IS;

          if (fl_otkata_shemy == NO)    dalvjaz_tek_wetka = 0;

          wywod_shemy(Sender);
          form_ls->SetFocus();
          form_ls->zagruzka_LSW(Sender);
}
//---------------------------------------------------------------------------

// i.30. задать признак элемента для строки при записи схемы в файл

void  zadatx_metku_stroki(int ii, char *stro)
{
          int l, i;
          char st[50];

          if (tek_alg.m_el[ii].ind == 0)  return;

          l = strlen(stro);
          while (l < GRANICA_N_EL) {  l++;  strcat(stro, " ");  }

          strcat(stro,"//.");

          itoa(tek_alg.m_el[ii].ind, st, 10);
          strcat(stro, st);

          if (tek_alg.m_el[ii].name[0] != 0) {

            strcat(stro, "  ");   strcat(stro, tek_alg.m_el[ii].name);
          }
```

```
          if (tek_alg.m_el[ii].n_bmp != -1) {

            for (i=0; i < RAZM_M_BITMAP; i++) {

              if (i == tek_alg.m_el[ii].n_bmp) {

                strcat(stro, "  ");   strcat(stro, m_imen_bmp[i]);
                break;
              }
            }
          }
}



// i.31. запись ВС в файл

void __fastcall Tform_dalvjaz::b_zapisxClick(TObject *Sender)
{
          FILE *fl1, *fl2;
          char stro[dl_stro], fname[80], str_sm[dl_stro], st[50];
          int ii, j, i, k, l, tek_sm;

          strcpy(fname, "shema.prg");
          if (fl_otkata_shemy == IS)   strcpy(fname, "shema.otk");

          write_comment(fname);

          if (fl_otkata_shemy == NO)  {////

          if (FileExists(AnsiString("shema.prg"))) {

           if (FileExists(AnsiString("shema.old")))   remove("shema.old");

           fl1 = fopen("shema.prg", "rt");
           fl2 = fopen("shema.old", "wt");
            while (fgets(stro, dl_stro-1, fl1) != NULL) {

              l = strlen(stro);
              stro[l-1] = 0;
              fprintf(fl2, "%s\n", stro);
            }
            fclose(fl1);
            fclose(fl2);
          }
          }////

          fl1 = fopen(fname, "wt");
          fprintf(fl1, "\n// %s\n\n", tek_alg.imq_zagol);

          tek_sm = L_GR_TEKSTA;   str_sm[0] = 0;
          for (i=0; i < L_GR_TEKSTA; i++)  strcat(str_sm, " ");

          for (j=0; j < (tek_alg.kol_wetok-1); j++) {

            ii = tek_alg.m_inewet[j];
            i=0;
            while ((ii < MAX_KOL_ELEM) && (i < tek_alg.m_el_wetki[j])) {

              if ((tek_alg.m_el[ii].used == IS) &&
                  (tek_alg.m_el[ii].wetka == (j+1))) {

                if ((tek_alg.m_el[ii].type == t_CE) ||
                    (tek_alg.m_el[ii].type == t_E)  ||
```

```
    (tek_alg.m_el[ii].type == t_HD) ||
    (tek_alg.m_el[ii].type == t_EI) ||
    (tek_alg.m_el[ii].type == t_EB))  {

  if (tek_sm > L_GR_TEKSTA)
                  {  tek_sm -= 2;   str_sm[tek_sm] = 0;  }
}

strcpy(stro, str_sm);

if ((tek_alg.m_el[ii].type == t_IF) ||
    (tek_alg.m_el[ii].type == t_EI) ||
    (tek_alg.m_el[ii].type == t_CE)) {

  if (tek_alg.m_el[ii].type == t_IF)  strcat(stro, "IF  ");
  if (tek_alg.m_el[ii].type == t_EI)  strcat(stro, "ELSE IF  ");
  if (tek_alg.m_el[ii].type == t_CE)  strcat(stro, "CE  ");

  if (tek_alg.m_el[ii].fl_usl == IS)  strcat(stro, "+");
  else                                strcat(stro, "-");

  zadatx_metku_stroki(ii,stro);   fprintf(fl1, "%s\n", stro);

  k=0;
  while (k < tek_alg.m_el[ii].kol_s_d) {

    strcpy(stro, str_sm);
    strcat(stro, tek_alg.m_el[ii].m_d[k].m);
    fprintf(fl1, "%s\n", stro);

    k++;
  }

  if (tek_alg.m_el[ii].type != t_CE) {
            tek_sm += 2;   strcat(str_sm, "  ");  }
}

if ((tek_alg.m_el[ii].type == t_HD) ||
    (tek_alg.m_el[ii].type == t_AD) ||
    (tek_alg.m_el[ii].type == t_CB)) {

  if (tek_alg.m_el[ii].type == t_HD) {
    strcat(stro, "HD  ");
    itoa(tek_alg.m_el[ii].wetka, st, 10);
    strcat(stro, st);
  }
  if (tek_alg.m_el[ii].type == t_AD) {
    strcat(stro, "AD  ");
    if (tek_alg.m_el[ii].wetka2 == 0)
      strcat(stro, "EXIT");
    else
    if (tek_alg.m_el[ii].wetka2 == -1)
      strcat(stro, "UNDEF");
    else {
      itoa(tek_alg.m_el[ii].wetka2, st, 10);
      strcat(stro, st);
    }
  }
  if (tek_alg.m_el[ii].type == t_CB)  strcat(stro, "CB");

  zadatx_metku_stroki(ii,stro);   fprintf(fl1, "%s\n", stro);

  if (tek_alg.m_el[ii].type == t_HD) {
    k=0;
```

```
            while (k < tek_alg.m_el[ii].kol_s_d) {

              strcpy(stro, str_sm);
              strcat(stro, tek_alg.m_el[ii].m_d[k].m);
              fprintf(fl1, "%s\n", stro);

              k++;
            }
          }

          if (tek_alg.m_el[ii].type != t_AD) {
                    tek_sm += 2;   strcat(str_sm, "  ");  }
        }

        if ((tek_alg.m_el[ii].type == t_EB) ||
            (tek_alg.m_el[ii].type == t_E)  ||
            (tek_alg.m_el[ii].type == t_RET)) {

          if (tek_alg.m_el[ii].type == t_EB)  strcat(stro, "ELSE");
          if (tek_alg.m_el[ii].type == t_E)   strcat(stro, "END");
          if (tek_alg.m_el[ii].type == t_CB)  strcat(stro, "RETURN");

          fprintf(fl1, "%s\n", stro);

          if ((tek_alg.m_el[ii].type == t_EB) ||
              (tek_alg.m_el[ii].type == t_EI))   {
                    tek_sm += 2;   strcat(str_sm, "  ");  }
        }

        if (tek_alg.m_el[ii].type == t_A) {

          if (tek_alg.m_el[ii].kol_s_d == 0) {
            zadatx_metku_stroki(ii,stro);   fprintf(fl1, "%s\n", stro);
          }
          else {
            k=0;
            while (k < tek_alg.m_el[ii].kol_s_d) {

              if (k == 0) {
                strcpy(stro, str_sm);
                strcat(stro, tek_alg.m_el[ii].m_d[k].m);
                zadatx_metku_stroki(ii,stro);
                fprintf(fl1, "%s\n", stro);
              }
              else {
                strcpy(stro, str_sm);
                strcat(stro, tek_alg.m_el[ii].m_d[k].m);
                fprintf(fl1, "%s\n", stro);
              }
              k++;
            }
          }
        }

        i++;
      }

      ii++;
    }

  }

  fprintf(fl1, "%s\n", "     HD  EXIT    //.");
  fprintf(fl1, "%s\n", "     RETURN      //.");
```

```
          fclose(fl1);


}
//---------------------------------------------------------------------------

// i.32. обработка для всплывающего информационного окна элемента

void __fastcall Tform_dalvjaz::wsplyw_inf_okno(int X, int Y, int j, int ii)
{
         int k, kc, x1, x2, y1, y2;

         if (tek_alg.m_el[ii].type == t_AD)  return;

         mouse_ind_el = -1;

         x1=dalvjaz_smx+ tek_alg.m_wert[j].x+ tek_alg.m_el[ii].x1;
         x2=dalvjaz_smx+ tek_alg.m_wert[j].x+ tek_alg.m_el[ii].x2;
         y1=dalvjaz_smy+ tek_alg.m_wert[j].y1+ tek_alg.m_el[ii].y1;
         y2=dalvjaz_smy+ tek_alg.m_wert[j].y1+ tek_alg.m_el[ii].y2;

         if ((X > x1) && (X < x2) && (Y > y1) && (Y < y2)) {

           if (wspl_fl_mouse == IS) {

             mouse_x = X;   mouse_y = Y;   mouse_ind_el = ii;
             return;
           }

           if (Panel4->Visible == false) {

             Panel4->Left = PaintBox1->Left + X + 30;
             Panel4->Top = PaintBox1->Top + Y + 20;

             if ((Panel4->Left + Panel4->Width) > (Width-20))
                       Panel4->Left = Panel4->Left - Panel4->Width;

             Panel4->Height = 57;

             Image1->Canvas->Font->Name = "Courier New";
             Image1->Canvas->Font->Size = 11;

             k = tek_alg.m_el[ii].kol_s_d;  if (k > 0)  k--;
             kc = tek_alg.m_el[ii].kol_s_k;
             if (tek_alg.m_el[ii].kol_s_d == 0)  if (kc > 0)  kc--;

             Panel4->Height=dalvjaz_chdy*(2+k+kc+1);
             Image1->Height=Panel4->Height;

             Image1->Canvas->Brush->Color= RGB(230,230,230);
             Image1->Canvas->FillRect
                (Rect(0,0,Image1->Width,Image1->Height));

             if ((tek_alg.m_el[ii].kol_s_d == 0) &&
                 (tek_alg.m_el[ii].kol_s_k == 0)) {
               Image1->Canvas->TextOut(5, 5,
                        //"12345678901234567890123456789012345678901234567890");
                        "Пусто");
             }
             else {
               y1 = 5;

               if (tek_alg.m_el[ii].kol_s_k > 0) {
```

```
                for (k=0; k < tek_alg.m_el[ii].kol_s_k; k++) {

                  Image1->Canvas->TextOut(
                                    5,y1,tek_alg.m_el[ii].m_k[k].m);
                  y1 += dalvjaz_chdy;
                }
              }

              if ((tek_alg.m_el[ii].kol_s_k > 0) &&
                  (tek_alg.m_el[ii].kol_s_d > 0)) {

                // черта между комментариями и данными

                form_dalvjaz->Image1->Canvas->MoveTo(0,y1+5);
                form_dalvjaz->Image1->Canvas->LineTo(Panel4->Width,y1+5);

                y1 += dalvjaz_chdy;
              }

              if (tek_alg.m_el[ii].kol_s_d > 0) {

                for (k=0; k < tek_alg.m_el[ii].kol_s_d; k++) {

                  Image1->Canvas->TextOut(
                                    5,y1,tek_alg.m_el[ii].m_d[k].m);
                  y1 += dalvjaz_chdy;
                }
              }
            }
            Panel4->Show();
          }
          wspl_inf_okno = IS;
        }
}


// i.33. обработка строки "да/нет" контекстного меню элемента

void __fastcall Tform_dalvjaz::mn_danetClick(TObject *Sender)
{
        if (tek_alg.m_el[mouse_ind_el].fl_usl == IS)
                    tek_alg.m_el[mouse_ind_el].fl_usl = NO;
        else        tek_alg.m_el[mouse_ind_el].fl_usl = IS;
        wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.34. обработка строки "редакция" контекстного меню элемента

void __fastcall Tform_dalvjaz::mn_redClick(TObject *Sender)
{
        form_red->zagruzitx_el(mouse_ind_el);
}
//---------------------------------------------------------------------------

// i.35. обработка для значка "копирование данных"

void __fastcall Tform_dalvjaz::mn_copyClick(TObject *Sender)
{
        tek_alg.m_el[mouse_ind_el].n_bmp = 0;
        wywod_shemy(Sender);
}
//---------------------------------------------------------------------------
```

```
// i.36. обработка для значка "вычислить"

void __fastcall Tform_dalvjaz::mn_calcClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 1;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.37. обработка для значка "обработка данных"

void __fastcall Tform_dalvjaz::mn_dataClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 2;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.38. обработка для значка "вывод на экран"

void __fastcall Tform_dalvjaz::mn_ekranClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 3;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.39. обработка для значка "сообщение об ошибке"

void __fastcall Tform_dalvjaz::mn_errClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 4;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.40. обработка для значка "проверка на ошибку"

void __fastcall Tform_dalvjaz::mn_iserrClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 5;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.41. обработка для значка "открыть файл данных"

void __fastcall Tform_dalvjaz::mn_otkrClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 6;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.42. обработка для значка "проверить данные"

void __fastcall Tform_dalvjaz::mn_prowdtClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 7;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------
```

```
// i.43. обработка для значка "получить внешние данные"

void __fastcall Tform_dalvjaz::mn_recdtClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 8;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.44. обработка для значка "сохранить файл данных"

void __fastcall Tform_dalvjaz::mn_zakrClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = 9;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.45. отмена значка для элемента

void __fastcall Tform_dalvjaz::mn_net_znachkaClick(TObject *Sender)
{
         tek_alg.m_el[mouse_ind_el].n_bmp = -1;
         wywod_shemy(Sender);
}
//---------------------------------------------------------------------------

// i.46 читать файл комментариев для элементов ВС

void __fastcall Tform_dalvjaz::read_comment(char *fn)
{
         FILE *fl;
         char fname[80], stro[dl_stro], *_u;
         int i, ii, l, fl_no_read, n, ne, kol;

         strcpy(fname, fn);
         l = strlen(fname);     if (l <= 4)  return;

         fname[l-4] = 0;
         if (fname[l-3] == 'p')    strcat(fname, ".txt");
         else                      strcat(fname, ".ott");

         fl_no_read = IS;

         fl=fopen(fname, "rt");
         if (fl == NULL)  return;
         while (fgets(stro, 199, fl) != NULL)  {
           l = strlen(stro);
           stro[l-1] = 0;   l--;

           if (str_probelow(stro) == IS)  continue;

           // начальная строка комментария, если точка после числа
           // в колонке 1, 2 или 3 (начиная с 0)

           if ((stro[1] == '.') || (stro[2] == '.') || (stro[3] == '.')) {

             i=0;
             while (stro[i] != ' ')  i++;   n = i+1;    _u = &stro[n];
             i--;
             stro[i] = 0;     ne = atoi(stro);

             if (ne != 0) {
```

```
                i=0;   kol = 0;
                while ((i < MAX_KOL_ELEM) && (kol < tek_alg.kol_elem)) {

                  if (tek_alg.m_el[i].used == IS) {

                    if (tek_alg.m_el[i].ind == ne) {

                      fl_no_read = NO;    tek_alg.m_el[i].kol_s_k = 1;

                      strcpy(tek_alg.m_el[i].m_k[0].m, _u);     ii = i;

                      break;
                    }
                    kol++;
                  }
                  i++;
                }
              }
              else   fl_no_read = IS; // непонятное пропускаем

            }
            else {
              if (fl_no_read == NO) {

                udal_probely(stro, n, NO);

                strcpy(tek_alg.m_el[ii].m_k[ tek_alg.m_el[i].kol_s_k ].m, stro);
                tek_alg.m_el[i].kol_s_k++;
              }
            }
          }
          fclose(fl);
}


// i.47 записать файл комментариев для элементов ВС

void __fastcall Tform_dalvjaz::write_comment(char *fn)
{
          FILE *fl;
          char fname[80], stro[dl_stro], st[50], *_u;
          int i, l, kol, n;

          strcpy(fname, fn);
          l = strlen(fname);     if (l <= 4)  return;

          fname[l-4] = 0;
          if (fname[l-3] == 'p')    strcat(fname, ".txt");
          else                      strcat(fname, ".ott");

          fl = fopen(fname, "wt");

          i=0;   kol = 0;
          while ((i < MAX_KOL_ELEM) && (kol < tek_alg.kol_elem)) {

            if (tek_alg.m_el[i].used == IS) {

              if ((tek_alg.m_el[i].ind != 0) &&
                  (tek_alg.m_el[i].kol_s_k != 0)) {

                itoa(tek_alg.m_el[i].ind, stro, 10);

                strcat(stro, ". ");   n = strlen(stro);
                st[0] = 0;
```

```
                for (l=0; l < n; l++)   strcat(st, " ");

                strcat(stro, tek_alg.m_el[i].m_k[0].m);
                fprintf(fl, "%s\n", stro);

                for (l=0; l < (tek_alg.m_el[i].kol_s_k-1); l++) {

                  strcpy(stro, st);
                  strcat(stro, tek_alg.m_el[i].m_k[l+1].m);
                  fprintf(fl, "%s\n", stro);
                }
                fprintf(fl, "%s\n", "");
              }
              kol++;
            }
            i++;
          }
          fclose(fl);
}
```

Литература, ссылки в сети Интернет и прочие материалы

1. В.Д. Пароннджанов «Как улучшить работу ума: Алгоритмы без программистов – это очень просто !» Москва, Издательство «Дело», 2001

2. В.Д. Пароннджанов «ЯЗЫК ДРАКОН краткое описание» http://store.oberoncore.ru/lib/book/DrakonDescription.rar

3. В.В. Фаронов «Основы Турбо Паскаля» Москва, Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992

4. Разделы "Визуальное программирование" и "Структурные редакторы" форума сайта oberoncore.ru

5. В качестве прототипов предлагаемых пиктограмм действий взяты стандартные пиктограммы прикладного ПО ОС MS Windows