**Введение**

Задачей дипломного проекта являлось изучение оценки состояния ЭЭС и концепций построения математического обеспечения информационно-вычислительных подсистем, знакомство с КП Компоновщик расчетных схем, освоение Windows‑программирования, приобретение навыков работы в интегрированной среде Developer Studio и разработка некоторых элементов пользовательского интерфейса Компоновщика расчетных схем в операционной системе Windows.

В первой главе была выбрана математическая модель режима, рассмотрены несколько критериев оценки состояния ЭЭС. Обоснован выбор метода обобщенной нормальной оценки для оценивания состояния ЭЭС, как обладающего существенными преимуществами по сравнению с другими критериями. В качестве численного метода принят в общем случае метод Ньютона-Рафсона. Наряду с этим методом, при наличии некоторых условий целесообразно применять метод Ньютона-Рафсона по параметру. Для решения систем линейных уравнений по итерационным формулам используется метод Гаусса (LU‑разложение). Вычисления производятся с учетом свойств разреженных матриц. Показано разнесение вычислений вне реального времени (на подготовительном этапе) и непосредственно в реальном времени.

Во второй главе рассматривается назначение и функциональные возможности программы Компоновщик расчетных схем. Приведен алгоритм формирования расчетной схемы, удовлетворяющий требованиям, предъявляемым к системам подготовки, отладки и поддержания данных.

В третьей главе описываются основные особенности Windows‑программирования: процесс создания главного окна программы, процедуры обработки сообщений. Показываются отличия модальных и немодальных окон диалога. Приводится описание панелей инструментов и состояния. Подробно показан процесс создания и работы строкового и оконного редакторов, реализация функций, предоставляемых пользователю.

В последней, четвертой, главе рассматривается вопрос из раздела техники безопасности на тему: «Разработка мероприятий по безопасной эксплуатации ПЭВМ».

**1. Оценка состояния**

**1.1 Постановка задачи**

Инвариантность поведения автоматизированной системы диспетчерского управления (АСДУ) относительно внешних возмущений, а, значит, обоснованность принятых решений и эффективность управлением режимом энергосистемы в реальном времени может быть обеспечена в принципе, если используется достоверная информация о параметрах режима и схеме электрических соединений, о составе и состоянии основного оборудования, о параметрах и характеристиках отдельных объектов, и т.д. Наиболее важным и трудным оказывается получение информации о текущих параметрах режима и схеме электрических соединений, требующей проведения измерений в реальном времени. Непосредственное измерение всех параметров режима невозможно, недостающая часть должна быть восстановлена расчетным путем. Низкая достоверность телеизмеряемой информации, возможные отказы каналов связи, ограниченные возможности информационно-измерительной сети затрудняют непосредственное воспроизведение физики явлений. Повысить достоверность телеизмерений (ТИ), восстановить расчетным путем недостающую часть параметров режима, воспроизвести физику явлений позволяют методы теории оценок, которые делятся на две группы: методы, учитывающие априорную информацию об оцениваемых параметрах режима; методы, не учитывающие такую информацию.

Установившийся режим электроэнергетической системы (ЭЭС) описывается системой нелинейных алгебраических уравнений

(1.1)



где x, у – соответственно оцениваемые и измеряемые параметры режима.

Форма записи (1.1) зависит от выбранной схемы замещения отдельных элементов и системы координат, разделения параметров режима на измеряемые и оцениваемые; конкретный состав системы – составом используемых ТИ. Разделение параметров режима на оцениваемые и измеряемые производится по смыслу решаемой задачи; руководствоваться следует лишь двумя требованиями: в качестве оцениваемых принимаются те, зная которые, легко рассчитать все остальные параметры режима без решения дополнительной системы уравнений; в конечном счете необходимо получить параметры режима, которые будут использованы в дальнейшем при оптимизации. По техническим соображениям, а также для потребностей диспетчерского персонала, измеряются потоки мощностей по ветвям и узлах, напряжения, токи в ветвях. Это и определяет выбор оцениваемых параметров режима – модули и фазы или действительные и мнимые части узловых напряжений.

Связь между оцениваемыми и измеряемыми параметрами режима

*y* = f(x) (1.2)

устанавливается уравнениями потокораспределения, конкретный вид которых определяется выбранной формой записи, в частности:

◊ для суммарной активной и реактивной мощности в i-ом узле

(1.3)



◊ для потоков мощностей в начале ветви (i, j)

(1.4)



где: n – число независимых узлов; – узловые напряжения (); – проводимость ветви (i, j) (); – проводимость на землю в узле i ().



После введения матричных обозначений:

,



мы и получаем (1.2).

В измерениях, проводимых в реальной ЭЭС, всегда присутствуют погрешности, обусловленные погрешностями измерительной аппаратуры, помехами в каналах связи, неодновременностью замеров, поэтому вектор измерений может быть представлен суммой вектора истинных значений *у* и вектора ошибок w



*у* + w (1.5)



Если предположить, что случайный вектор ошибок распределен по нормальному закону

(1.6)



где R – ковариационная матрица ошибок измерений; m – число измерений, то наиболее правдоподобными считаются те значения w, при которых достигается максимум плотности распределения (используется метод максимального правдоподобия, основанный на максимизации функции (1.6), называемой функцией правдоподобия. Эта показательная функция достигает своего максимума, когда ее показатель минимален [1]).

В общем случае в качестве меры близости измеренных параметров режима и их расчетных значений f(x) может быть выбран критерий



(1.7)



Задача оценки состояния ЭЭС сводится к нахождению такого вектора оцениваемых параметров режима, который доставляет значения измеряемым, близкие к измеренным в смысле выбранного критерия (1.7)

Оценка состояния ЭЭС – сложный процесс, в котором можно выделить ряд основных этапов:

1. Выбор математической модели режима.
2. Построение критерия оценки.
3. Разработка численного метода и алгоритма оценивания.

Деление носит условный характер, все этапы взаимосвязаны: свойства математической модели режима являются определяющими при построении критерия оценивания, алгоритм оценивания должен учитывать как свойства выбранного критерия близости, так и математической модели режима. В конечном счете, алгоритм оценивания должен удовлетворять следующим требованиям: высокая скорость и надежность сходимости, высокая точность и устойчивость результатов, высокое быстродействие, незначительный объем требуемой оперативной памяти ЭВМ.

Рассмотрим подробнее каждый из этапов.

**1.2 Математическая модель режима**

Если в начале каждой ветви дерева измеряется поток активной и реактивной мощности, то соответствующая математическая модель режима имеет единственное решение в целом. Любая модель режима, полученная из указанной добавлением новых измеряемых параметров режима, также имеет единственное решение [1].

Модель режима называется корректной, если [1]:

1. для любого *y* существует решение (условие разрешимости);
2. решение является единственным (условие однозначности);
3. решение непрерывно зависит от *у* (условие устойчивости).

При нарушении хотя бы одного из этих условий задача оценки состояния называется некорректно поставленной (математическая модель называется некорректно поставленной).

**1.3 Критерий оценки**

На выбор критерия оценки влияют: характер исследуемого процесса (статический или динамический), наличие или отсутствие статистических данных об измеряемых и оцениваемых параметрах, корректная или некорректная постановка задачи. Определяющей является некорректная постановка задачи оценивания состояния ЭЭС. Она проявляется в том, что сколь угодно малые погрешности измерений приводят к сколь угодно большим ошибкам в результатах оценки. Метод максимального правдоподобия дает один из наиболее распространенных критериев.

(1.8)



причем оценка состояния ЭС сводится к нахождению

(1.9)



Метод, основанный на (1.9) (метод взвешенных наименьших квадратов), позволяет получить такую оценку, которая доставляет значения измеряемым параметрам режима, близкие к измеренным в смысле минимума дисперсии измерений.

Если матрица ковариации ошибок измерений R неизвестна или ее получение затруднено, вместо (1.8) используется

(1.10)



и оценка находится из условия достижения

(1.11)



Метод, основанный на (1.11) (метод наименьших квадратов), позволяет получить такую оценку , которая доставляет значения измеряемым параметрам режима, близкие к измеренным в смысле минимума суммы квадратов невязок.



Если система нелинейных алгебраических уравнений совместна, то решения (1.9.) и (1.11) совпадают. Для переопределенных и несовместных нелинейных алгебраических уравнений решение (1.9) и (1.11), вообще говоря, не совпадают: решение (1.9) зависит от выбора матрицы ковариаций.

Некорректность математической модели режима отражается на свойствах этих критериев, характере решения задачи оценивания:

1. нарушено требование однозначности – критерий оценки является многоэкстремальной функцией; каждое решение модели режима – это точка экстремума критерия оценки;
2. нарушено требование разрешимости – выполняется необходимое условие существования минимума, ранг матрицы частных производных понижается;



1. нарушено требование непрерывности либо критерий оценки не имеет минимума в области определения (в целом), либо выполняется необходимое условие существования минимума, ранг матрицы частных производных понижается.



Использование рассмотренных методов с учетом некорректности постановки задачи оценивания состояния ЭЭС становится проблематичным.

Наличие априорных данных об оцениваемых параметрах и матрице ковариации ошибок задания априорных данных S позволяет использовать критерий



(1.12)



и получить оценку из условия достижения

(1.13)



Метод, реализующий (1.13) (байесова оценка), в ряде случаев позволяет локализовать нужное решение за счет использования априорных данных.

Каждый из рассмотренных методов имеет свои недостатки и достоинства. Общим недостатком является невозможность использования для оценки состояния ЭЭС с учетом ее некорректной постановки.

Для решения некорректно поставленных задач был предложен метод регуляризации

α>0,



где – сглаживающая или регуляризующая функция; – стабилизирующая функция; – параметр регуляризации.



Идея метода основана на использовании априорных сведений об оцениваемых параметрах: физический смысл имеют только ограниченные решения.

Проведенные исследования [1] показали:

1. если решение математической модели режима является неоднозначным, то локализовать нужное (действительное) не всегда удается;
2. возможны случаи, когда итерационный процесс решения (1.8) затягивается – в стабилизирующую функцию входят несоизмеримые по величине параметры режима, и стремление ограничить решение приводит к чрезмерному сглаживанию;
3. целесообразность задания априори параметра регуляризации.

Недостатки метода регуляризации могут быть устранены после соответствующей его модификации.

**1.4 Обобщенная нормальная оценка**

Этот метод соединяет в себе достоинства метода наименьших квадратов, байесовой оценки, метода регуляризации и дает возможность решать задачу в ее некорректной постановке, обеспечивая устойчивость вычислительного процесса и позволяя получать решение, наиболее близкое к истинному режиму ЭЭС. Сущность метода обобщенной нормальной оценки состоит в следующем.

К оценке состояния ЭЭС можно подойти с позиций решения системы нелинейных алгебраических уравнений

(1.14)



где: m – количество измеряемых параметров режима; n+1 – общее число узлов ЭЭС.

Если известны точные значения измеряемых параметров режима *у*, то решение x математической модели режима (1.14) существует; оно может быть единственным или неединственным (в последнем случае нужное решение локализуется после согласования области определения и области значений) [2].

Если известны приближенные значения правых частей (1.14)

(1.15)



где w – вектор случайных величин с математическим ожиданием М[w]=0, то для данной математической модели режима в пределах заданного уровня погрешности измерений существует целый класс режимов, для каждого из которых решение

(1.16)



может существовать (быть единственным или неединственным) или не существовать, а сколь угодно малые изменения измеряемых параметров могут приводить к сколь угодно большим изменениям решения [2]. По существу f отображает множество различных решений в пространстве оцениваемых параметров в неразличимое множество измерений в пространстве наблюдений.

Для некорректной модели режима требуется уточнить понятие «решение». Среди множества решений (1.15) естественно выбрать наиболее близкое к априорным данным и одновременно доставляющее измеряемым параметрам режима значения, близкие к измеренным . Если выбрать в качестве меры близости евклидову длину вектора, то этим требованиям отвечает решение, доставляющее минимум



, . (1.17)



Первое слагаемое (аналог обобщенного решения) характеризует близость измеренных и расчетных f(x) значений, второе слагаемое (аналог нормального решения) – близость априорных данных и решения x. Назначение параметра регуляризации – согласование меры близости в пространстве оцениваемых параметров и меры близости в пространстве наблюдений (косвенно решается проблема согласования области определения и области значений).



Решение, доставляющее минимум (1.17), называется обобщенным нормальным решением, а метод, реализующий этот критерий, – методом обобщенной нормальной оценки (МОНО).

Параметр регуляризации обобщенно учитывает статистические свойства измерений и априорных данных, его значение задается априори как



где: - дисперсия измерений; - дисперсия задания априорных данных.



При таком выборе параметра регуляризации МОНО дает неухудшающуюся, устойчивую к погрешности измерений и к изменениям параметра регуляризации оценку, а верхняя норма матрицы ковариации ошибок оценки оказывается минимальной.

В качестве априорной информации, используемой при оценке состояния реальной ЭЭС, можно использовать:

1) результаты предыдущей оценки;

1. измеренные значения напряжений (их номинальные значения); ограниченность фаз узловых напряжений (δ → 0).

Второй случай менее благоприятен. Часть априорных данных (например, измеренные напряжения) принадлежит области определения, другая часть (например, фазы узловых напряжений) может и не принадлежать к ним. Достоверность таких данных различна, полученная оценка параметра регуляризации находится в широком диапазоне (10÷105) [2]. Целесообразно для каждой группы априорных данных ввести свои весовые коэффициенты:

а) CU1 – для измеренных напряжений;

б) CU2 – для номинальных напряжений (если измерений не проводилось);

в) Сδ – для фаз узловых напряжений.

Тогда критерий оценки перепишется в виде

,



где: – диагональная матрица с вышеуказанными весовыми коэффициентами, – априорные данные (для фаз узловых напряжений это значения на к-ой итерации).



Для реальных ЭЭС: CU1 =10-2, CU2 =10-4, Cδ =1, и диапазон изменения параметра регуляризации сужается: 103<<105 [2]



**1.5 Численные методы решения**

Принимая во внимание все выше сказанное, в конечном счете задача оценивания состояния ЭЭС сводится к решению экстремальной задачи

(1.18)



по итерационной формуле

, (1.19)



где: k – номер итерации; – направление продвижения на (к+1) – ой итерации из точки хк; – коэффициент, определяющий длину шага в направлении ; – приращение на к-ой итерации; начальное приближение задается.



В результате решения (1.19) будет получена последовательность с определенными свойствами.



Для выбранной модели режима и построенного критерия оценки эффективность алгоритма оценки состояния ЭЭС определяется свойствами численного метода решения (1.19) и характеризуется такими критериями, как: скорость и надежность сходимости, точность решения, время счета, сложность алгоритма, требуемый объем оперативной памяти ЭВМ и т.д.

Численные методы решения (1.19) используют ту или иную аппроксимацию либо целевой функции

(1.20)



либо вектор-функции f(x). Наибольшее распространение получил метод Ньютона-Рафсона, в котором используется разложение в ряд Тейлора нелинейной вектор-функции f(x) в окрестности произвольной точки хк до членов первого порядка малости включительно

f (x) = f (xk) + fx (xk) (x – xk). (1.21)

Подстановка (1.21) в (1.20) дает:



Из необходимого условия минимума следует:

,



тогда приращение на к-ой итерации находится

,



где нижний индекс указывает, по какому вектор-аргументу осуществляется дифференцирование; x – x k = Δx k; x, x k – достаточно близкие точки.

Итерационный процесс (1.19) продолжается до достижения заданной точности расчетов ε:

⏐Δ x k⏐ ≤ ε.

Для уменьшения времени счета проверку можно производить только для модулей узловых напряжений.

Наличие стабилизирующей функции позволяет получить решение независимо от начального приближения, итерационный процесс сходится за две-четыре итерации, а число итераций в основном определяется качеством ТИ и «тяжестью» режима [2].

Оценка, вообще говоря, зависит от параметра регуляризации α. При завышенных значениях α возможно появление т.н. эффекта сглаживания, который может быть ослаблен, если воспользоваться следующим подходом.

Пусть на к-ом шаге методом Ньютона-Рафсона получена оценка хК и приращение ΔхК. Величина шага в направлении ΔхК может быть выбрана из условия достижения минимума суммы квадратов небалансов мощностей, т.е.



Приравняв к нулю и выразив из этого равенства , получим



.



Итерационный процесс, реализованный по формуле

, (1.22)



продолжается до тех пор, пока не будет нарушено условие

,



где характеризует скорость уменьшения суммы квадратов небалансов мощностей (обычно принимается равной 0.99).



Метод Ньютона-Рафсона по параметру целесообразно использовать в двух случаях:

а) когда имеются точные значения измеряемых параметров режима у;

б) когда возникают затруднения с оценкой числового значения .



Учитывая вышеперечисленные достоинства метода обобщенной нормальной оценки, естественно будет использовать его в дальнейшем для оценки состояния ЭЭС.

**1.6 Вычислительные аспекты**

Специфические особенности ЭЭС и МОНО играют решающую роль в рациональной организации вычислительного процесса.

Используемые при оценке состояния ЭЭС матрицы – матрица узловых проводимостей, матрица частных производных, матрица коэффициентов системы линейных алгебраических уравнений

(1.23)



содержат незначительное число ненулевых элементов, т.е. являются разреженными: значительного сокращения времени счета и существенной экономии используемого объема оперативной памяти ЭВМ можно добиться, если хранить ненулевые элементы и оперировать с ними.

Память, используемая для хранения разреженных матриц, состоит из двух частей: основной, содержащей числовые значения, и накладной, предназначенной для хранения информации о местоположении в матрице хранимых значений. Чем сложнее схема хранения, тем больше накладная память и меньше основная, и наоборот. Время доступа к числовым значениям и, следовательно, время счета зависит также от схемы хранения. Процесс вычислений при статичной схеме хранения, эффективный в смысле требований к памяти и времени счета, может потребовать катастрофических накладных расходов при динамичном изменении схемы хранения. Из вышесказанного следует, что схему хранения желательно выбирать с учетом процесса вычислений.

Для решения систем линейных алгебраических уравнений вида (1.23)

(1.24)



( , )



используется метод Гаусса или его модификации. В методе Гаусса система уравнений (1.24) решается в два хода – прямой и обратный. При прямом ходе матрица коэффициентов приводится к верхней треугольной форме. Для этого к системе (1.24) с t неизвестными применяется (t -1) – шаговый процесс исключения неизвестных. В результате на (t -1) – ом шаге будет получена треугольная система:

(1.25)



Обратный ход метода Гаусса состоит в последовательном вычислении неизвестных из (1.25), начиная с последнего уравнения.

Рассмотренные преобразования удобно реализовать в матричном виде. Если обозначить матрицу коэффициентов (1.25)

(1.26)



и ввести матрицу преобразований наr – том шаге

(1.27)



то

. (1.28)



Операция обращения матрицы преобразования (1.27) равносильна инвертированию недиагональных элементов, а произведение нижних треугольных матриц дает такую же матрицу, поэтому

(1.29)



где

(1.30)



Выражение (1.29) – т. н.LU – разложение матрицыА в виде произведения нижней треугольной матрицы L и верхней треугольной матрицы U.

Замена z=Uh показывает, что h можно получить, решая треугольные системы:

Lz=b (1.31)

Uh=z (1.32)

Выражение (1.31) – матричная запись заключительной части прямого хода метода Гаусса (пересчета свободных членов), а (1.32) – матричная запись обратного хода. Для симметричной матрицы



где D – диагональная матрица с элементами

i=1,2…., t,



разложение

(1.33)



называется – разложением.



Допущение относительно диагональных элементов (), называемых главными, существенно. В противном случае для обеспечения численной устойчивости необходима та или иная форма выбора главного элемента, т.е. перестановки строк и (или) столбцов. Эти перестановки определяются в процессе решения системы уравнений путем компромисса между требованиями численной устойчивости и сохранением разреженности. Для разреженных матриц общего вида нельзя установить порядок исключения неизвестных, пока не начались собственно вычисления. Более того, такой выбор главного элемента может привести к крайне нежелательному росту числа ненулевых элементов.



Одно из основных достоинств МОНО состоит в том, что гауссово исключение не требует выбора главных элементов для поддержания численной устойчивости. Это означает, что матрицу коэффициентов можно переупорядочить, не заботясь о численной устойчивости, причем до начала численного решения: выбирается такая последовательность исключения неизвестных, которая приводит к появлению минимального числа ненулевых элементов. Еще одна важная особенность такого выбора исключаемой переменной состоит в симметричном переупорядоченииматрицы коэффициентов – имеет место симметричная перестановка строк и столбцов [1].



Отмеченные особенности, присущие только МОНО именно в силу самого выбора параметра регуляризации, имеют далеко идущие практические последствия. Если порядок исключения неизвестных не зависит от результатов реального процесса вычислений, то наиболее трудоемкая часть расчетов, связанная с формированием структуры начального заполнения матрицы коэффициентов, ее упорядочением, резервированием места для новых ненулевых элементов, появляющихся в процессе реальных вычислений, может и должна выполняться вне реального времени на подготовительном этапе. Схема хранения должна обеспечивать высокую эффективность вычислений в реальном времени, оставаясь при этом статичной. В реальном времени реализуются вычисления, связанные с формированием и решением системы уравнений.

Т.к. матрица коэффициентов симметрична, достаточно пересчитывать и хранить только ее верхнюю треугольную часть. Если для каждой строки имеется список столбцов с ненулевыми элементами, то он полностью определяет, в каких строках элементы каких столбцов пересчитываются. Для удобства поиска в этом списке индексы столбцов желательно располагать в порядке возрастания. Например, если на r‑ом шаге в r‑ой строке ненулевые элементы находятся в столбцах r, s, q, то пересчитываются коэффициенты в s‑ой (в столбцах s и q) и в q‑ой (в столбце q) строках.

В матрице частных производных каждому i‑му узлу соответствует два столбца 2i‑1, 2i, а в матрице коэффициентов А – блочная матрица второго порядка:

.



Измерению ветви (i, j) соответствуют четыре ненулевые блочные матрицы: Ai i,Ai j, Aj i, Aj j (i < j). .

Для каждого такого блока местоположение всех четырех элементов однозначно определяется номером строки и номером столбца блока, что равносильно указанию места установки измерительного датчика. Аналогично, номер узла и список смежных с ним узлов определяют блоки ненулевых элементов для измерения в r‑ом узле.

Элементы матрицы коэффициентов хранятся блоками по строкам. Для каждого блока ненулевых элементов номер столбца указывается в массиве «индексы столбцов». Местоположение первого блока каждой строки задается в массиве «указатель индексов строк» [1]. Блочное представление дает существенную экономию памяти как при хранении, так и при формировании системы уравнений. В действительности кодировка расстановки ТИ непосредственно определяет местоположение блоков ненулевых элементов в схеме хранения, следовательно, отпадает необходимость запоминания промежуточных результатов (матрицы частных производных).

Т.о., все необходимые предпосылки для рациональной организации вычислительного процесса гарантируются МОНО. Наиболее трудоемкая часть расчетов должна выполняться вне реального времени на подготовительном этапе. К ним относятся:

1. Формирование структуры первоначального заполнения матрицы коэффициентов. Она (структура) однозначно определяется расстановкой ТИ и топологией электрической сети; формируется с учетом всех ТИ для типовой схемы электрических соединений, в которой все объекты, оснащенные устройствами телесигнализации (ТС), считаются включенными. Текущее состояние объектов, не оснащенных устройствами ТС, отражается в исходной схеме электрических соединений. На подготовительном этапе резервируется место для всех возможных ненулевых элементов. Сформированная таким образом структура заполнения и, следовательно, схема ее хранения может использоваться при оценивании состояния ЭЭС в реальном времени с любым составом ТИ и при любых производимых в сети коммутациях, не приводящих к появлению новых узлов: отключение части ТИ и (или) ветвей отражается только на числовых значениях элементов матрицы.
2. Упорядочение – определение последовательности исключения неизвестных. Это равносильно перенумерации узлов расчетной схемы. Предпочтительней такая последовательность исключения неизвестных, которая приводит к появлению минимального числа новых ненулевых элементов. Наиболее часто используются два алгоритма динамического упорядочения. В первом из них на каждом шаге метода Гаусса исключается неизвестная, соответствующая строка которой содержит минимальное число ненулевых элементов (если таких неизвестных несколько, то выбор произволен), во втором – неизвестная, исключение которой приводит к появлению наименьшего числа новых ненулевых элементов. Оба алгоритма дают достаточно близкие результаты, но первый алгоритм динамического упорядочения предпочтительней, так как проще, требует меньших затрат времени и памяти.
3. Имитация исключения Гаусса с резервированием места под новые ненулевые элементы и формирование схемы хранения матрицы коэффициентов. Структура первоначального заполнения (верхняя треугольная часть) запоминается блоками по строкам. С учетом установленной последовательности исключения неизвестных резервируется место для новых ненулевых элементов.
4. Формирование матрицы узловых проводимостей. Для ускорения процесса поиска нужного элемента ненулевые элементы этой матрицы, несмотря на симметричность, хранятся полностью. Схема хранения аналогична рассмотренной ранее.

В реальном времени выполняются вычисления, необходимые собственно для оценивания ЭЭС:

1. Ввод текущих ТИ.
2. Формирование системы уравнений (1.24).
3. Решение системы уравнений (1.24).
4. Реализация одного шага итерационного процесса (1.19).
5. Проверка критерия окончания счета . Если условие не выполняется, перейти к п. 2.



1. Расчет потокораспределения по результатам оценивания.

Разнесение расчетов во времени существенно упрощает алгоритм оценивания состояния ЭЭС. Возможности, заложенные на подготовительном этапе, определяют как эффективность вычислений в реальном времени в смысле экономии памяти и уменьшения времени счета, так и особенности численной реализации:

а) хранение и обработка только верхней треугольной части матрицы коэффициентов системы уравнений (1.24);

б) блочное хранение ненулевых элементов;

в) использование кодировки расстановки ТИ, непосредственно определяющей местоположение блоков ненулевых элементов;

г) вычисление в неявном виде матрицы частных производных без запоминания промежуточных результатов;

д) использование статичной схемы хранения.

Выделение подготовительного этапа, который будет называться формированием расчетной схемы, – это основная концепция построения математического обеспечения информационно-вычислительной подсистемы. Такой подход, поощряя раздельное программирование отдельных задач и их этапов, с одной стороны, максимально упрощает программы решения задач реального времени, с другой стороны, позволяет формировать расчетную схему, пригодную для решения как можно большего числа задач [2].

**1.7 Концепции построения математического обеспечения СПУРТ**

Математическое обеспечение (МО) – это совокупность баз данных, программное обеспечение (ПО) и математическая модель электрической системы вместе с кодировкой расстановки ТИ, ТС и схемами хранения используемых разреженных матриц.

Распределение исходных данных между подсистемами оперативно-информационного управляющего комплекса (ОИУК), их подготовка и хранение организуются таким образом, чтобы максимально облегчить работу пользователя, наиболее полно использовать имеющуюся в информационно-управляющей подсистеме (ИУП) нормативно-справочную информацию и построить адаптивную математическую модель режима в реальном времени, и, кроме того, ввести *жесткую* адресацию результатов расчетов для отображения их на дисплеях с использованием форматов. Связь между исходными данными различных ЭВМ, исходными данными и данными, отображаемыми на различные виды терминалов, устанавливается с помощью таблицы соответствия, которая формируется на ЭВМ информационно-вычислительной подсистемы (ИВП) на подготовительном этапе. На этом этапе выполняется наиболее трудоемкая часть расчетов, не требующая расчетов в реальном времени: обслуживание баз данных, выбор и просмотр произвольной информации о расстановке ТИ и ТС, проверка правильности подготовки исходных данных, реализация принципов оптимального упорядочивания, формирование расчетной схемы, определение местоположения ненулевых элементов и упаковка разреженных матриц [3].

Имеющаяся на ЭВМ ИУП нормативно-справочная информация дополняется кодировкой расстановки ТИ и ТС, а на ЭВМ ИВП создается единая для МО база данных. Способы задания отдельных элементов, представления схем замещения и схем электрических соединений достаточно гибки и универсальны и допускают:

◊ упрощенное и детальное представление схемы замещения;

◊ упрощенное и детальное представление схем электрических соединений;

◊ любые сочетания представлений схемы замещения и схемы электрических соединений расчетного узла;

◊ физическое и мнимое удаление элементов из базы данных (элемент не учитывается при формировании расчетной схемы, но сохраняется в базе данных);

◊ установку измерительного датчика в произвольной точке схемы замещения до (после) поперечной проводимости ветви.

Однозначное соответствие между данными различных ЭВМ обеспечивается кодировкой расстановки ТИ, ТС и использованием «позиционности». Указанное соответствие не меняется (оно может дополняться) при вводе новых объектов и устройств телемеханики.

Программная проверка правильности подготовки исходных данных (представление чисел, соответствие последовательности чисел определенному элементу расчетной схемы, допустимые отношения и предельные значения параметров системы и режима, соответствие классов напряжений, связности графа электрической сети, расстановка ТИ по условию полноты математической модели режима, соответствие кодировки расстановки ТИ, ТС и расчетной схемы, связность узла расчетной схемы по исходным значениям ТС, контроль текущих размерностей массивов и т.п.) осуществляется автоматически при формировании эталонной расчетной схемы [3].

Расстановка ТИ должна выбираться из условия существования и единственности решения математической модели режима ЭЭС. Несмотря на ограниченность числа ТИ, возможные их потери, модель режима, по крайней мере, должна быть полной. С этой целью предусмотрено использование априорных данных, данных режимного дня, эксплуатационных замеров. Указанные данные могут вводиться как вручную (оперативно, в реальном времени), так и автоматически (программно).

База данных рассматривается как автономная, локальная, открытая система, допускающая поэтапное накапливание и расширение исходных данных по мере освоения задач и подключения новых комплексов программ. МО независимо от структуры и содержания базы данных, т. к. используются две базы данных: основная (форматные записи) и рабочая (бесформатные записи). Из основной базы данных в рабочую копируется необходимая информация [3].

Достигнут разумный компромисс между противоречивыми требованиями сокращения времени счета и уменьшением используемой оперативной памяти за счет:

◊ разнесения вычислений как по времени, так и между отдельными программными продуктами;

◊ использования статичных схем хранения разреженных матриц;

◊ тесной увязки кодировки расстановки ТИ и ТС и блочных схем хранения, применения единой математической модели ЭС.

Такой подход позволил формировать расчетную схему ЭС вне реального времени (на подготовительном этапе).

Отдельные компоненты ПО взаимодействуют через единую модель электрической системы, подстраиваемую под внешние условия за счет:

◊ коррекции кодировки расстановки ТИ и ТС, текущих значений ТИ, ТС и параметров элементов электрической системы в реальном времени (адаптивный контур);

◊ оперативной коррекции вручную расстановки ТИ. значений ТС и параметров элементов электрической системы;

◊ ручной (вне реального времени) коррекции данных на любом уровне предусмотренной иерархии исходных данных.

Эталонные расчетные схемы, учитывающие ввод новых объектов и (или) устройств телемеханики, а также требующие ручной коррекции расстановки ТИ, формируются заблаговременно.

Предусмотрена возможность одновременного хранения десяти эталонных расчетных схем, отражающих наиболее характерные схемы электрических соединений. С этой целью создана база расчетных схем, из которой при решении предусмотренных задач одна из эталонных схем (рабочая) копируется в рабочие файлы.

Для моделирования, анализа и хранения режимов создана база режимов (до 12 режимов). Предусмотрена возможность записи произвольного режима, являющегося результатом решения одной из задач, в базу режимов.

Все расчеты, включая и формирование отображаемых на дисплеях кадров, производятся на ЭВМ ИВП. В ИВП передаются текущие ТИ и ТС циклически или (и) спорадически, информация о местоположении устройств телемеханики посылается только по запросу, в обратном направлении передаются отображаемые на дисплеях кадры [3].

Таким образом, математическое обеспечение ИВП удовлетворяет жестким требованиям, характерным для задач реального времени.

Наиболее трудоемкая часть расчетов, не требующая вычислений в реальном времени, реализована в КП Компоновщик расчетных схем.

**2. Комплекс программ компоновщик расчетных схем**

**2.1 Назначение комплекса программ. Компоновщик расчетных схем**

Компоновщик расчетных схем – сервис-оболочка баз данных математического обеспечения, предназначенная для облегчения и ускорения процесса подготовки и отладки исходных данных, создания и поддержания основных баз данных, реализации всех трудоемких расчетов, не связанных с вычислениями в реальном времени, в частности, формировании (компоновки) эталонных расчетных схем.

КП поддерживает единый стиль диалога с пользователем, имеет достаточный набор подсказок и подменю, существенно упрощающих его освоение и работу. Такие возможности, как комбинирование параметров в любом сочетании, совместный анализ параметров различных элементов, классификация параметров по группам, программное обнаружение ошибок, автоматический поиск источника ошибки значительно облегчают наиболее тяжелый и трудоемкий процесс – подготовку и отладку исходных данных.

Компоновщик является базовым комплексом МО СПУРТ, обеспечивает взаимную увязку всех данных ИУП и ИВП, формирует единую математическую модель электрической системы для всех решаемых технологических задач.

**2.2 Алгоритм формирования расчетной схемы**

Исходные данные формируются из различных источников, которыми могут быть ИУП (расстановка ТИ, ТС, значения ТИ, ТС) или, в перспективе, банк данных (параметры системы). Каждый источник может создаваться и поддерживаться независимо и самостоятельно.

При подготовке, отладке и, в особенности, поддержании исходных данных возникает серьезное противоречие: с одной стороны, желательно иметь надежные, практически неизменные исходные данные, с другой стороны, математическая модель электрической системы должна быть достаточно гибкой, легко и оперативно подстраиваемой под изменившиеся условия (в частности, при изменениях схемы электрических соединений и априори задаваемых мощностей узлов, не оснащенных устройствами телемеханики).

Возможность независимого и самостоятельного создания и поддержания любого уровня установленной иерархии исходных данных (рис. 2.1) решает данную проблему: основная база данных должна содержать полный набор исходных данных, и изменения вносятся по мере введения новых объектов энергосистемы и устройств телемеханики; рабочая база данных должна отражать оперативное состояние объектов энергосистемы, не оснащенных устройствами телемеханики.

Источники данных

Основная база данных

Рабочая база данных

Рис. 2.1. Иерархия исходных данных

При таком подходе изменения, вносимые в источники данных, могут автоматически переноситься в базы данных; изменения в основной базе данных – в рабочую базу данных. Вместе с тем любой уровень этой иерархии может создаваться и поддерживаться независимо и самостоятельно.

Сформулированным требованиям отвечает алгоритм формирования расчетной схемы, реализованный по схеме, которая изображена на рис. 2.2. Порядок выполнения алгоритма зависит от того, какая задача решается: первоначальная подготовка и отладка исходных данных или внесение изменений на разных уровнях иерархии исходных данных.

Источники

данных

Основная

база данных

Рабочая

база данных

База

расчетных схем

Утилиты

Трансляция

Компоновка

Отладка

Редактор

Рис. 2.2. Структурная схема алгоритма формирования расчетной схемы

1. Передать из ИУП расстановку ТИ.
2. Передать из ИУП расстановку ТС.
3. Передать из ИУП текущие значения ТИ и ТС или данные из архива.
4. Подготовить параметры узлов с помощью Редактора.
5. Транслировать параметры узлов.
6. Просмотреть Протокол: если обнаружены ошибки, перейти к п. 4; в противном случае выполнить п. 7.
7. Подготовить параметры ветвей с помощью Редактора.
8. Транслировать параметры ветвей.
9. Просмотреть Протокол: если обнаружены ошибки, перейти к п. 7; в противном случае выполнить п. 10.
10. Отредактировать расстановку ТИ.
11. Транслировать файл расстановки ТИ основной базы данных.
12. Просмотреть Протокол: если обнаружены ошибки, перейти к п. 10; в противном случае выполнить п. 13.
13. Отредактировать расстановку ТС.
14. Транслировать файл расстановки ТС основной базы данных.
15. Просмотреть Протокол: если обнаружены ошибки, перейти к п. 13; в противном случае выполнить п. 16.
16. Отредактировать значения ТИ и ТС (соответствующий файл транслируется автоматически).
17. Подготовить с помощью Отладчика:
18. управляющую информацию;
19. вспомогательные массивы.

18. Если необходимо, изменить с помощью Отладчика:

1. параметры узлов;
2. параметры ветвей;
3. расстановку ТИ;
4. расстановку ТС.
5. Сформировать (скомпоновать) расчетную схему.
6. Просмотреть Протокол: если обнаружены ошибки, то либо перейти к п. 4 (если необходимо изменить и основную базу данных), либо перейти к п. 7 (если изменяется только рабочая база данных); в противном случае выполнить п. 21.
7. Получить необходимые выходные документы.
8. Завершить работу.

Все функции КП Компоновщик расчетных схем, позволяющие ему работать по приведенному алгоритму, можно разбить на пять основных групп: трансляция, отладка, компоновка, утилиты для работы с файлами и вывод выходных документов. Рассмотрим их более подробно.

**2.3 Основные функции КП Компоновщик расчетных схем**

Главное меню Компоновщика расчетных схем включает:

◊ копирование, обмен файлами с ИУП, просмотр файлов (Утилиты);

◊ подготовку исходных данных, поддержание и отладку основной базы данных (Трансляция);

◊ создание, поддержание и отладку рабочей базы данных (Отладка);

◊ формирование расчетных схем и поддержание базы расчетных схем (Компоновка);

◊ вывод выходных документов (Документы).

Подробное описание меню и подменю приводится ниже.

**2.3.1 Утилиты**

С помощью утилит выполняются вспомогательные функции: копирование, обмен файлами с ИУП, просмотр файлов.

**Копирование.** Расчетная схема с выбранным номером копируется из базы расчетных схем в рабочие файлы (становится рабочей).

**Прием ТИ и ТС из ИУП.** Данные могут быть как архивными, так и текущими. Для архивных данных ТИ (ТС передаются только для текущего времени) запрашивается дата и время. Разница между текущим и указанным временем не должна превышать 24 часа. При успешном обмене на ПЭВМ создается файл с заданной спецификацией.

**Прием РТИ из ИУП.** После запроса ИУП предлагается указать спецификацию файла. При успешном обмене на ПЭВМ создается файл с указанной спецификацией.

**Прием РТС из ИУП.** После запроса ИУП предлагается указать спецификацию файла. При успешном обмене на ПЭВМ создается файл с указанной спецификацией.

**Просмотр файлов.** Вспомогательная функция, предназначенная для оперативного визуального контроля файлов (предусмотрен запрос спецификации просматриваемого файла). Рекомендуется использовать для проверки файлового обмена и выходных документов.

**2.3.2 Трансляция**

Назначение – создание, поддержание и отладка основной базы данных. Файлы основной базы данных могут создаваться либо самостоятельно и независимо с помощью встроенного Редактора, либо с помощью утилит. Редактор также позволяет поддерживать основную базу данных, обрабатывать протоколы трансляции и компоновки.

В процессе трансляции файлы основной базы данных преобразуются в файлы рабочей базы данных. Сообщения об обнаруженных ошибках заносятся в Протокол, содержимое которого после трансляции любого файла или компоновки расчетной схемы полностью обновляется.

В подменю «Трансляция» предлагается сначала выбрать тип исходных данных: параметры узлов, параметры ветвей, расстановка ТИ, расстановка ТС, значения ТИ, значения ТС. Затем выбрать режим работы: трансляция, Редактор, Протокол (Протокол доступен, если он создан).

**Параметры узлов.** Параметры узлов могут создаваться и поддерживаться с помощью Редактора независимо и самостоятельно.

Каждому узлу соответствует строка, состоящая из нескольких полей. Назначение каждого поля поясняется титрами.

В процессе редактирования параметров узлов для пользователя доступны вызовы функций, предоставляющих следующие возможности:

◊ Помощь.

◊ Локальная помощь. Выводится подсказка, поясняющая назначение каждого поля, какие параметры вводятся и их условные обозначения, единицы измерения (если это необходимо).

◊ Вызов Протокола.

◊ Титры. В ряде случаев, в особенности при отладке, необходимо редактировать лишь определенную комбинацию параметров. Кроме того, для удобства желательно задавать параметры в любой комбинации и в любой последовательности. Все эти возможности реализуются данной функцией.

◊ Поиск. Данная функция облегчает и ускоряет поиск нужного параметра.

◊ Редактирование названий узлов.

Параметры узлов редактируются в оперативной памяти, предусмотрен запрос на запись.

**Параметры ветвей.** Параметры ветвей могут создаваться и поддерживаться с помощью Редактора независимо и самостоятельно.

Каждой ветви соответствует строка, состоящая из нескольких полей. Назначение каждого поля поясняется титрами.

В процессе редактирования параметров ветвей для пользователя доступны вызовы тех же функций, что и при редактировании параметров узлов (см. «Параметры узлов»).

Параметры ветвей редактируются в оперативной памяти, предусмотрен запрос на запись.

**Расстановка ТИ.** Файл расстановки ТИ может быть создан с помощью утилит, если нормативно-справочная информация ТИ ИУП дополнена соответствующей кодировкой расстановки. Редактор позволяет создавать и поддерживать этот файл независимо и самостоятельно.

Каждому ТИ отводится несколько полей. Назначение каждого поля поясняется титрами (приведены условные обозначения).

В процессе редактирования расстановки ТИ для пользователя доступны вызовы тех же функций, что и при редактировании параметров узлов (см. «Параметры узлов»).

Расстановка ТИ редактируется в оперативной памяти, предусмотрен запрос на запись.

**Расстановка ТС.** Файл расстановки ТС может быть создан с помощью утилит, если нормативно-справочная информация ТС ИУП дополнена соответствующей кодировкой расстановки. Редактор позволяет создавать и поддерживать этот файл независимо и самостоятельно.

Каждой ТС отводится несколько полей. Назначение каждого поля поясняется титрами (приведены условные обозначения).

В процессе редактирования расстановки ТС для пользователя доступны вызовы тех же функций, что и при редактировании параметров узлов (см. «Параметры узлов»).

Расстановка ТС редактируется в оперативной памяти, предусмотрен запрос на запись.

**Значения ТИ.** Редактируются значения ТИ, переданные из ИУП с помощью утилит. Такая необходимость возникает когда либо часть ТИ заведомо ошибочна, либо часть ТИ отключена, либо результаты измерений нужно согласовать с данными из диспетчерской ведомости, в частности при обработке данных режимного дня.

Каждому ТИ соответствует строка, состоящая из нескольких полей (указывается кодировка расстановки ТИ и измеренное значение). Назначение полей поясняется титрами (приведены условные обозначения).

В процессе редактирования значений ТИ для пользователя доступны вызовы тех же функций, что и при редактировании параметров узлов (см. «Параметры узлов»), за исключением вызова Протокола.

Значения ТИ редактируются в оперативной памяти, причем для редактирования доступны только измеренные значения, и транслируются автоматически (т.е. использовать режим трансляции необязательно). Предусмотрен запрос на запись.

**Значения ТС.** Редактируются значения ТС, переданные из ИУП с помощью утилит (для того, чтобы передать значения ТС из ИУП, необходимо при обмене файлами запросить текущие данные). Такая необходимость возникает когда либо часть ТС заведомо ошибочна, либо часть датчиков ТС отключена, либо значения ТС нужно согласовать с текущей схемой электрических соединений.

Каждой ТС соответствует строка, состоящая из нескольких полей (указывается кодировка расстановки ТС, исходное и текущее значение ТС). Назначение полей поясняется титрами (приведены условные обозначения).

В процессе редактирования значений ТС для пользователя доступны вызовы тех же функций, что и при редактировании параметров узлов (см. «Параметры узлов»), за исключением вызова Протокола.

Значения ТС редактируются в оперативной памяти, причем для редактирования доступны только текущие значения, и транслируются автоматически (т.е. использовать режим трансляции необязательно). Предусмотрен запрос на запись.

**2.3.3 Отладка**

Назначение – создание, поддержание и отладка рабочей базы данных. Файлы рабочей базы данных могут создаваться независимо и самостоятельно с помощью Отладчика или после трансляции соответствующих файлов основной базы данных. Отладчик позволяет также поддерживать рабочую базу данных, обрабатывать как протокол трансляции, так и протокол компоновки.

Предусмотрен широкий набор функций, облегчающих и ускоряющих отладку. В частности, реализованы различные возможности сортировки, комбинирования, поиска, совместного анализа исходных данных. Полностью автоматизирован поиск источника ошибки.

В подменю «Отладка» предлагается выбрать тип исходных данных: управляющая информация, параметры узлов, параметры ветвей, вспомогательные массивы, расстановка ТИ, расстановка ТС.

Отлаженные исходные данные могут быть записаны в рабочую базу данных с возвратом в главное меню (функция «Сохранить»). При выходе из меню отладки, если отлаженные данные не были сохранены, последует запрос на запись.

**Параметры узлов.** Параметры узлов могут создаваться и поддерживаться с помощью Отладчика независимо и самостоятельно.

Каждому узлу соответствует строка, состоящая из нескольких полей. Назначение каждого поля поясняется титрами (приведены условные обозначения).

Для облегчения и ускорения отладки параметров узлов предлагаются различные возможности сортировки, комбинирования, поиска, совместного анализа исходных данных. Параметры узлов можно сортировать в порядке возрастания номеров узлов.

В процессе отладки параметров узлов для пользователя доступны вызовы функций, предоставляющих следующие возможности:

◊ Редактирование названий узлов.

◊ Вызов Протокола. Протокол доступен либо после трансляции любого файла основной базы данных, либо после компоновки расчетной схемы.

◊ Помощь.

◊ Поиск по номеру узла. Данная функция облегчает и ускоряет поиск нужного узла по его номеру; одновременно проверяется наличие дубля.

◊ Поиск по номеру отображения узла. Данная функция облегчает и ускоряет поиск нужного узла по его номеру отображения; одновременно проверяется наличие дубля.

◊ Список мнимо удаленных узлов. Функция избавляет от необходимости поиска мнимо удаленных узлов и облегчает удаление и включение узлов при формировании расчетной схемы.

◊ Априорные данные. Функция эффективна при оперативном изменении как состава узлов с априорными данными, так и самих априорных данных.

◊ Связи. Наибольшие трудности вызывает поиск и устранение ошибок, обусловленных несоответствием параметров узлов и ветвей. Функция позволяет анализировать фрагмент электрической сети, для чего по определенному алгоритму формируется список узлов, связанных с указанным. Соответствие параметров выявленных таким образом связей и параметров ветвей, примыкающих к указанному узлу, можно проверить визуально, воспользовавшись соответствующей функцией Отладчика.

◊ Справка. Предоставляется информация об использованных номерах отображений узлов, что позволяет избежать возможное их дублирование.

**Параметры ветвей.** Параметры ветвей могут создаваться и поддерживаться с помощью Отладчика независимо и самостоятельно.

Каждой ветви соответствует строка, состоящая из нескольких полей. Назначение каждого поля поясняется титрами (приведены условные обозначения).

Для облегчения и ускорения отладки параметров ветвей предлагаются различные возможности сортировки, комбинирования и поиска, совместного анализа исходных данных. Параметры ветвей можно сортировать в порядке возрастания номеров инцидентных узлов.

В процессе отладки параметров ветвей для пользователя доступны вызовы функций, предоставляющих следующие возможности:

◊ Вызов Протокола. Протокол доступен либо после трансляции любого файла основной базы данных, либо после компоновки расчетной схемы.

◊ Помощь.

◊ Узел. Одна из возможностей выделения группы ветвей (выбираются ветви, примыкающие к указанному узлу).

◊ Ветвь. Данная функция облегчает и ускоряет поиск нужной ветви по ее номеру; одновременно проверяется наличие дубля.

◊ Поиск мнимо удаленных ветвей. Функция избавляет от необходимости поиска мнимо удаленных ветвей и облегчает удаление и включение ветвей при формировании расчетной схемы.

◊ Трансформаторы. Функция предназначена для отладки параметров трансформаторных ветвей.

◊ Реакторы. Функция предназначена для отладки параметров реакторов.

◊ Справка. Предоставляется информация об использованных номерах ветвей, что позволяет избежать возможного их дублирования.

**Расстановка ТИ.** Файл расстановки ТИ рабочей базы данных создается после трансляции файла расстановки ТИ основной базы данных. Отладчик позволяет создавать, поддерживать и отлаживать этот файл независимо и самостоятельно.

Для облегчения и ускорения отладки расстановки ТИ предлагаются различные возможности сортировки, комбинирования, поиска, совмещения с фрагментом электрической сети. ТИ можно сортировать в порядке возрастания их номеров.

В процессе отладки расстановки ТИ для пользователя доступны вызовы функций, предоставляющих следующие возможности:

◊ Вызов Протокола. Протокол доступен либо после трансляции любого файла основной базы данных, либо после компоновки расчетной схемы.

◊ Помощь.

◊ Узел. Одна из возможностей выделения группы ТИ: выбираются ТИ, датчики которых установлены возле (в) указанного узла; одновременно выводится фрагмент электрической сети (список ветвей, примыкающих к указанному узлу), что значительно облегчает проверку соответствия кодировки расстановки ТИ и расчетной схемы.

◊ Ветвь. Функция облегчает и ускоряет поиск датчиков ТИ, установленных на ветви.

◊ Список мнимо удаленных ТИ. Функция избавляет от поиска мнимо удаленных ТИ и облегчает отключение и включение датчиков ТИ при формировании расчетной схемы.

◊ Поиск. Данная функция облегчает и ускоряет поиск нужного ТИ по его номеру; одновременно проверяется наличие дубля.

◊ Код. Автоматически восстанавливается код измеряемого параметра режима ранее мнимо удаленного ТИ независимо от срока давности.

**Расстановка ТС.** Файл расстановки ТС рабочей базы данных создается после трансляции файла расстановки ТС основной базы данных. Отладчик позволяет создавать, поддерживать и отлаживать этот файл независимо и самостоятельно.

Для облегчения и ускорения отладки расстановки ТС предлагаются различные возможности сортировки, комбинирования, поиска, совмещения с фрагментом электрической сети. ТС можно сортировать в порядке возрастания их номеров.

В процессе отладки расстановки ТС для пользователя доступны вызовы функций, предоставляющих следующие возможности:

◊ Вызов Протокола. Протокол доступен либо после трансляции любого файла основной базы данных, либо после компоновки расчетной схемы.

◊ Помощь.

◊ Узел. Одна из возможностей выделения группы коммутационных аппаратов (КА): выбираются КА, относящиеся к указанному узлу; одновременно выводится фрагмент электрической сети (список ветвей, примыкающих к указанному узлу), что значительно облегчает проверку соответствия кодировки расстановки ТС и расчетной схемы.

◊ Список мнимо удаленных ТС. Функция избавляет от поиска мнимо удаленных ТС и облегчает отключение и включение датчиков ТC при формировании расчетной схемы.

◊ Поиск. Данная функция облегчает и ускоряет поиск нужной ТС по ее номеру; одновременно проверяется наличие дубля.

◊ Код. Автоматически восстанавливается код ранее мнимо удаленной ТС независимо от срока давности.

**Управляющая информация.** Управляющая информация создается и поддерживается только Отладчиком. Эта информация отражает специфические особенности реализованных алгоритмов (по умолчанию для большинства параметров берутся рекомендуемые значения).

В процессе редактирования управляющей информации для пользователя доступны вызовы функций, предоставляющих следующие возможности:

◊ Переключение «окон». Весовые коэффициенты, используемые при идентификации параметров элементов электрической системы и характеризующие достоверность априорных значений идентифицируемых параметров, выделены в отдельную группу, для их просмотра и коррекции предоставляется отдельное «окно». Переход из одного «окна» в другое осуществляется с помощью этой функции.

◊ Вызов Протокола. Протокол доступен либо после трансляции любого файла основной базы данных, либо после компоновки расчетной схемы.

**Вспомогательные массивы.** Вспомогательные массивы создаются и поддерживаются только Отладчиком. Эти массивы (генераторные узлы, внешние узлы (связи), автотрансформаторы с РПН) используются при формировании выходных форм ИУП, поэтому должна выдерживаться жесткая позиционная привязка; любое их изменение должно быть согласовано с выходными формами ИУП. Объекты, указанные в этих массивах, могут быть отключены (не учитываться) в формируемой расчетной схеме.

В процессе редактирования вспомогательных массивов для пользователя доступны вызовы функций, предоставляющих следующие возможности:

◊ Переключение «окон». Выбирается нужный вспомогательный массив.

◊ Вызов Протокола. Протокол доступен либо после трансляции любого файла основной базы данных, либо после компоновки расчетной схемы.

◊ Помощь.

◊ Поиск. Функция облегчает и ускоряет поиск нужного элемента вспомогательного массива, одновременно проверяется наличие дубля.

**2.3.4 Компоновка**

Компонуется расчетная схема из отдельных оттранслированных файлов – формируется единая для всех решаемых технологических задач математическая модель электрической системы.

На этом этапе производится основная программная проверка правильности подготовки отдельных типов исходных данных (представление чисел, соответствие последовательности чисел определенному элементу, допустимые отношения и предельные значения параметров системы и режима, текущие размерности массивов), а также их соответствие как составляющих единой математической модели (соответствие классов напряжений, связность графа электрической сети, соответствие кодировки расстановки ТИ и ТС расчетной схеме, топологическая наблюдаемость энергосистемы, связность узлов расчетной схемы по исходным значениям ТС). Сообщения об обнаруженных ошибках заносятся в Протокол.

В процессе компоновки расчетной схемы выполняется наиболее трудоемкая часть расчетов, не требующая вычислений в реальном времени (реализация принципов оптимального упорядочения; формирование собственно математической модели электрической системы; имитация исключений Гаусса, определение местоположения ненулевых элементов и упаковка разреженных матриц; формирование и упаковка вспомогательных массивов, обеспечивающих высокую вычислительную эффективность реализованных в СП СПУРТ алгоритмов).

Подменю «Компоновка» содержит два пункта: Компоновка, Протокол. Протокол доступен либо после трансляции любого файла основной базы данных, либо после компоновки расчетной схемы.

Если выбран пункт «Компоновка», предоставляется справка о сформированных расчетных схемах Следует выбрать номер формируемой расчетной схемы и ввести шифр для нее.

Основные этапы компоновки расчетной схемы контролируются, для чего на экран дисплея выводится их перечень и отмечается характер завершения каждого из них.

**2.3.5 Документы**

Выходные документы выводятся в файл, который создается после выбора подменю «Документы».

Выходные документы: таблица соответствия, каталог кадров, параметры системы, расстановка ТИ, расстановка ТС, идентифицируемые элементы. Все они, за исключением параметров системы и расстановки ТИ, могут выдаваться для любой существующей расчетной схемы.

◊ Таблица соответствия устанавливает жесткую позиционную связь между данными, которыми обмениваются ИУП и ИВП.

◊ Каталог кадров содержит шаблоны с указанием адресов для отображения обобщенной информации.

Выходные документы сопровождаются необходимыми пояснениями.

Пользовательский интерфейс КП Компоновщик расчетных схем разработан таким образом, чтобы облегчить работу пользователя, а использование при программировании стандартных элементов интерфейса ОС Windows должно обеспечить единый стиль диалога с пользователем.

**3. Особенности windows–программирования**

Windows 95 – одна из последних версий графической операционной системы (ОС) Windows, представленной впервые в ноябре 1985 года для использования на компьютерах типа IBM PC и совместимых с ним. За последнее десятилетие ОС Windows почти полностью вытеснила всех конкурентов и стала фактически эталоном ОС для персональных компьютеров.

В Windows 95 объединены средства, имеющиеся в более ранних ОС. К ним также добавлены новые средства, позволяющие более быстро выполнять большие объемы работ.

Программирование под Windows является одним из видов объектно-ориентированного программирования (ООП). В основе ОС Windows лежит ее способность отображать информацию, используя для этого окна. Окно – это прямоугольная область на экране, оно получает информацию от клавиатуры или мыши пользователя и выводит графическую информацию на своей поверхности (можно одновременно открыть несколько окон, чтобы одновременно работать с несколькими приложениями, тем самым увеличив производительность компьютера). Пользователь рассматривает окна на экране в качестве объектов и непосредственно взаимодействует с этими объектами.

Самой примечательной особенностью Windows является графический интерфейс пользователя. Дисплей сам становится источником, откуда в машину вводится информация. Он показывает различные графические объекты в виде картинок и конструкций для ввода информации (таких, как кнопки или полосы прокрутки). Используя клавиатуру или мышь, пользователь может непосредственно манипулировать этими объектами на экране. Графический интерфейс пользователя Windows существенно облегчает диалог с персональным компьютером.

ОС Windows позволяет создавать т.н. «дочерние окна управления». Дочернее окно обрабатывает сообщения мыши и клавиатуры и извещает родительское окно о том, что состояние дочернего окна изменилось. Оно инкапсулирует особые действия, связанные с графическим представлением окна на экране, реакцией на пользовательский ввод, и извещения другого окна при вводе важной информации. Можно создавать свои собственные дочерние окна управления, но есть также возможность использовать преимущества нескольких уже определенных классов окна (и оконных процедур), с помощью которых программа может создавать стандартные дочерние окна управления. Windows содержит следующие предопределенные классы окон: кнопка, радиопереключатель, флажок, окно редактирования, окно списка, окно комбинированного списка, статическое окно, полоса прокрутки.

Для упрощения создания Windows‑программ с интерфейсом пользователя, соответствующим интерфейсу оболочки системы, корпорация Microsoft разработала библиотеку органов управления общего пользования (common control library). Она содержит 17 элементов, которые условно можно разделить на четыре категории: органы управления главного окна (панель инструментов, окно подсказки, строка состояния), составные диалоговые органы управления (страница свойств, набор страниц свойств), органы управления Windows Explorer (дерево просмотра, список просмотра) и другие органы управления (например, горячая клавиша, индикатор процесса, окно с движком для выбора значения из диапазона). Как дочерние окна управления, так и органы управления общего пользования часто используются при создании Windows‑приложений.

Важной особенностью OC Windows является многозадачность. Под Windows 95 любая программа становится резидентной и несколько программ одновременно могут иметь вывод на экран и выполняться. Кроме того, программы сами по себе могут иметь несколько потоков выполнения, которые, как кажется, выполняются параллельно. В основе этого принципа лежит использование ОС аппаратного таймера для распределения процессорного времени между одновременно выполняемыми процессами. [4]

К основным преимуществам ОС Windows 95 относятся следующие:

◊ улучшенный интерфейс;

◊ более простое управление файлами, включая поддержку сетевых связей и длинных имен файлов;

◊ технология Plug and Play автоматически определяет и настраивает дополнительные устройства, подключаемые к компьютеру;

◊ 32 – битная многозадачная среда позволяет одновременно работать нескольким программам;

◊ улучшенные средства поиска;

◊ улучшенная поддержка мультимедиа;

◊ расширенные возможности установления связей, включая электронную почту, факсы, электронные доски объявлений, Internet;

◊ управление процессом печати документов.

Рассмотрим более подробно процесс создания прикладной программы Windows.

**3.1 Главное окно**

Прикладная программа Windows – это любая программа, которая специально написана для работы в операционной среде Windows и использующая интерфейс прикладной программы (API).

Прикладная программа Windows имеет следующие основные компоненты: основную функцию ***WinMain***; т.н. «оконную процедуру» – функцию ***WndProc***.

***WinMain*** – это точка входа в программу. В любой программе для Windows имеется функция ***WinMain***, она определяется следующим образом:

*integer function* ***WinMain*** *(hInstance, hPrevInstance, &*

*lpszCmdLine, nCmdShow).*

*hInstance* – дескриптор (описатель, хэндл) экземпляра. Это уникальное число, идентифицирующее программу, когда она работает под Windows;

*hPrevInstance* – дескриптор предыдущего экземпляра программы. Под Windows 95 этот параметр всегда NULL;

*lpszCmdLine* – указатель на оканчивающуюся нулем строку, в которой содержатся любые параметры, переданные в программу из командной строки.

*nCmdShow* – число, показывающее, каким должно быть выведено на экран окно в начальный момент.

В большинстве случаев эта функция делает следующее:

◊ вызывает функции инициализации, которые регистрируют классы окон, создают окна и выполняют другие инициализации;

◊ запускает цикл обработки сообщений из очереди прикладной программы;

◊ завершает работу программы.

***WndProc*** – это функция многократного вызова, т.е. функция программы, которую вызывает Windows. Сама программа никогда не вызывает эту функцию напрямую. Каждое окно имеет соответствующую оконную процедуру. Оконная процедура – это способ инкапсулирования кода, отвечающего за ввод информации и за вывод информации на экран, что реализуется путем посылки «сообщений» окну.

В процессе создания главного окна программы условно можно выделить четыре этапа:

1. регистрация класса главного окна;
2. создание главного окна;
3. отображение и корректировка главного окна;
4. организация цикла сообщений.

Рассмотрим подробнее каждый из этапов.

Регистрация класса окна.

Окно всегда создается на основе класса окна. Класс окна идентифицирует оконную процедуру, которая выполняет процесс обработки сообщений, поступающих окну, а также определяет некоторые атрибуты окна, уникальные для него, такие, например, как форма курсора и имя меню окна. На основе одного класса окна можно создать несколько окон.

Перед созданием окна для программы необходимо зарегистрировать класс окна путем вызова функции ***RеgisterСlаss***:

*ret =* ***RеgisterСlаss*** *(lpwc).*

*ret* – целочисленная переменная, содержащая возвращаемое функцией значение;

*lpwc* – указатель на структуру типа *T\_WNDCLASS*, содержащую в своих двенадцати полях информацию об имени, атрибутах, ресурсах и функции окна данного класса.

Заполнение структуры и регистрация класса окна производятся в ***WinMain***. Если регистрация прошла успешно, функция возвращает число, идентифицирующее регистрируемый класс. В противном случае она возвращает ноль.

Создание окна.

Окно создается с помощью функции ***СreateWindowEx*** (она отличается от аналогичной функции ***CreateWindow*** возможностью задания «расширенного» стиля окна и наличием параметра, используемого в качестве указателя на какие-то данные, на которые программа в дальнейшем могла бы ссылаться). Эта функция побуждает Windows создать окно, которое имеет указанный тип и принадлежит к определенному классу. Вызов функции ***СreateWindowEx*** осуществляется следующим образом:

*hWnd =* ***CreateWindowEx*** (&

*INT4 (WS\_EX\_DLGMODALFRAME),* ! «расширенный» стиль окна &

*lpszClassName,* ! Имя класса окна &

*lpszAppName,* ! Заголовок главного окна &

*INT4 (WS\_OVERLAPPEDWINDOW),* ! Стиль окна &

*0,* ! X – координата левого верхнего угла &

*0,*  ! Y – координата верхнего левого угла &

*GetSystemMetrics (SM\_CXSCREEN) – 32,* ! Ширина окна &

*GetSystemMetrics (SM\_CYSCREEN) – 32,* ! Высота окна &

*NULL,* ! Дескриптор родительского окна &

*hMenu,* ! Дескриптор меню окна &

*hInstance,*! Дескриптор экземпляра &

*NULL)* ! Параметры создания

Параметр с комментарием «имя класса окна» содержит строку, являющуюся именем только что зарегистрированного класса окна («Krs\_w»). Т.о., этот параметр связывает окно с классом окна.

«Заголовок главного окна» содержит текст, который появляется в строке заголовка («Компоновщик расчетных схем»).

Окно, созданное программой, является обычным перекрывающимся окном с заголовком, системным меню, главным меню, иконками для сворачивания, разворачивания и закрытия окна справа на строке заголовка и рамкой окна (стандартный стиль окон *WS\_ОVЕRLАРРЕDWINDOW*). Из предлагаемого ОС Windows набора «расширенных» стилей выбран стиль *WS\_EX\_DLGMODALFRAME*, добавляющий окну двойную рамку.

Параметр с комментарием «дескриптор родительского окна» устанавливается в NULL, т. к. у нашего окна отсутствует родительское окно. Параметр *hMenu* содержит дескриптор меню окна. В параметр *hInstance* помещается дескриптор экземпляра программы, переданный программе в качестве параметра функции ***WinMain***. Параметр с комментарием «параметры создания» при необходимости используется в качестве указателя на какие-нибудь данные, на которые программа в дальнейшем могла бы ссылаться (в рассматриваемой программе такие данные не используются – параметр установлен в NULL).

Вызов ***СreateWindowEx*** возвращает дескриптор созданного окна. Этот дескриптор хранится в переменной *hWnd*. Для многих функций Windows в качестве параметра требуется *hWnd*, указывающий Windows, к какому окну применить функцию. Если функция ***СreateWindowEx*** не может создать окно, она возвращает NULL.

Отображение и корректировка окна.

Хотя функция ***СreateWindowEx*** и создает окно, она автоматически не индицирует его. Программа должна предусмотреть индикацию окна с помощью функции ***ShowWindow*** и корректировку области пользователя окна с помощью функции ***UpdateWindow***.

Функция ***ShowWindow*** побуждает Windows индицировать новое окно:

*bret =* ***ShowWindow*** *(hWnd, nCmdShow)*

*hWnd* – дескриптор созданного функцией ***CreateWindowEx*** окна;

*nCmdShow* – величина, передаваемая в качестве параметра функции ***WinMain*** и задающая начальный вид окна на экране;

*bret* – логическая переменная, содержащая возвращаемое функцией значение.

Функция ***ShowWindow*** выводит окно на экран. Вызов функции ***UpdateWindow*** вызывает затем перерисовку рабочей области:

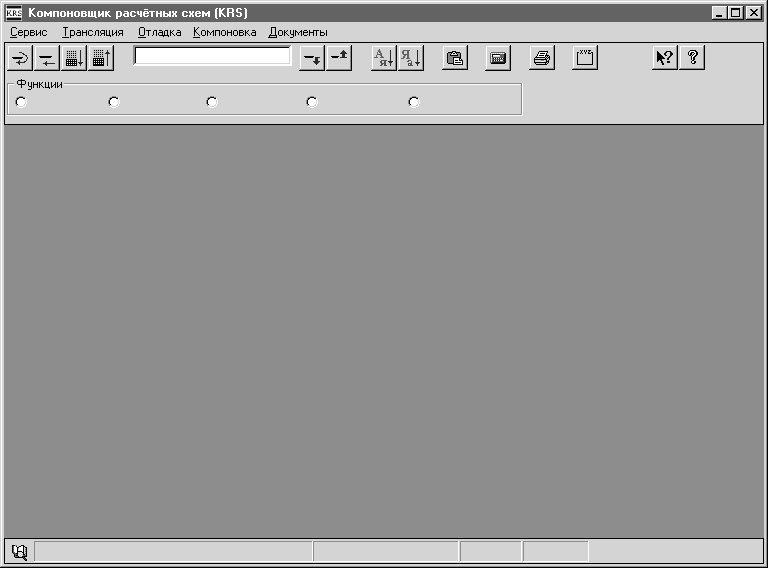
*bret =* ***UpdateWindow*** *(hWnd).*

При обработке этого вызова в оконную процедуру посылается сообщение *WМ\_РАINТ*.

Обе функции при успешном завершении возвращают TRUE, в противном случае – FALSE.

Вид главного окна Компоновщика расчетных схем представлен на рисунке 3.1.

Рис. 3.1. Вид главного окна



Как видно из рис. 3.1, главное окно Компоновщика содержит заголовок, строчку меню (структура которого была рассмотрена во 2‑ой главе), рабочую область, а также панель инструментов и панель состояния, которые более подробно будут рассмотрены ниже.

Цикл обработки сообщений

Итак, окно окончательно выведено на экран. Теперь программа должна подготовить себя для получения информации от пользователя через клавиатуру и мышь. Windows поддерживает «очередь сообщений» для каждой программы, работающей в данный момент в системе Windows. При вводе информации Windows преобразует ее в «сообщение», которое помещается в очередь сообщений программы.

Цикл обработки сообщений – это программный цикл, при работе в котором функция ***WinMain*** извлекает сообщения из очереди сообщений и направляет их соответствующим окнам. Для рассматриваемой программы цикл обработки сообщений выглядит следующим образом:

*do while (****GetMessage*** *(mesgt, NULL, 0, 0))*! Получить сообщение

*i = 0*

*if (hDlgModeLess. NE. NULL) then* ! Диалоговое окно активно?

*if (****IsDialogMessage*** *(hDlgModeLess, mesgt)) i =1*! Есть сообщение

! для немодального диалогового окна

*end if*

………! Аналогичным образом распределяются

………! сообщения для других немодальных

………! диалоговых окон, создаваемых программой

*if (i.ne.0) cycle*

*i =* ***TranslateMessage*** *(mesgt)* ! Транслировать сообщение

*i =* ***DispatchMessage*** *(mesgt)* ! Отправить сообщение

*end do*

Вызов функции ***GetMessage***, с которого начинается цикл обработки сообщений, извлекает сообщение из очереди сообщений. Этот вызов передает Windows указатель на структуру *mesgt* типа *T\_МSG*. Второй, третий и четвертый параметры, установленные в NULL или 0, показывают, что программа получает все сообщения от всех окон, созданных этой программой. Windows заполняет поля структуры сообщений информацией об очередном сообщении из очереди сообщений. Наиболее важными полями этой структуры являются:

◊ *hWnd* – дескриптор окна, для которого предназначено сообщение;

◊ *message* – тип сообщения;

◊ *wParam, lParam* – два 32‑разрядных параметра сообщения, смысл и значение которых зависят от особенностей сообщения.

Если поле message сообщения, извлеченного из очереди сообщений, равно любому значению, кроме *WМ\_QUIТ*, то функция ***GetMessage*** возвращает TRUE. Сообщение *WМ\_QUIТ* заставляет программу прервать цикл обработки сообщений.

При организации цикла обработки сообщений программы, создающей в ходе работы немодальные окна диалога[[1]](#footnote-1), необходимо учитывать, что сообщения для немодальных окон диалога проходят через очередь сообщений программы. Цикл обработки сообщений должен передавать такие сообщения соответствующим оконным процедурам окон диалога. Это реализуется следующим образом. Функция создания немодального диалога ***CreateDialog*** возвращает дескриптор окна (не равный 0), который запоминается в глобальной переменной (например*, hDlgModeLess*). Если диалог создан и сообщение предназначено для него, то функция ***IsDialogMessage*** отправляет сообщение оконной процедуре окна диалога, возвращая TRUE, тогда вспомогательная переменная *i* принимает значение 1, сигнализируя тем самым о перехвате сообщения диалоговым окном. В противном случае функция возвращает FALSE и переменная *i* не меняется. Если созданы и другие активные немодальные диалоги, то для них реализуются аналогичные операции. Таким образом гарантируется, что функции ***TranslateMessage*** и ***DispatchMessage*** будут вызываться только тогда, когда немодальные окна диалога не созданы или нет сообщений для них.

Т.к. программе необходимо обрабатывать вводимые с клавиатуры символы, нужно преобразовывать каждое полученное сообщение с помощью функции ***TranslateMessage***. Эта функция отыскивает пары сообщений *WM\_KEYDOWN* и *WM\_KEYUP* и генерирует соответствующее сообщение *WM\_CHAR*.

Функция ***DispatchMessage*** отправляет сообщение для его обработки соответствующей оконной процедуре. В рассматриваемой программе такой оконной процедурой является функция ***MainWndProc***. После того, как ***MainWndProc*** обработает сообщение, оно возвращается в Windows, которая все еще обслуживает вызов функции ***DispatchMessage***. Windows возвращает управление в программу к следующему за вызовом ***DispatchMessage*** оператору и цикл обработки сообщений в очередной раз возобновляет работу, вызывая ***GetMessage***.

Реальная работа начинается в оконной процедуре, которая обеспечивает реакцию на вводимую информацию и сообщения системы управления окнами, получаемые от Windows.

**3.2 Процедура обработки сообщений**

Оконная процедура определяется следующим образом:

*integer function* ***MainWndProc*** *(hWnd, mesg, wParam, lParam).*

Первым параметром является *hWnd*, дескриптор получающего сообщение окна. Второй параметр – число, которое идентифицирует сообщение. Параметры *wParam* и *lParam* содержат дополнительную информацию о сообщении и называются «параметрами сообщения». Конкретное значение этих параметров определяется типом сообщения.

Обработка сообщений

Оконная процедура получает сообщения от Windows. Это могут быть сообщения о вводе, переданные функцией ***WinMain*** (например, *WM\_KEYDOWN, WM\_KEYUP, WM\_MOUSEMOVE, WM\_TIMER*), или сообщения системы управления окнами, поступающие непосредственно от Windows в обход очереди прикладной программы или цикла обработки сообщений (например, *WM\_CREATE, WM\_DESTROY, WM\_PAINT*). Каждое получаемое окном сообщение идентифицируется номером, который содержится в параметре *mesg* оконной процедуры. В модуле MSFWINTY.F90 Fortran PowerStation 4.0 определены именованные константы, начинающиеся с префикса *WМ* для каждого типа сообщений.

Обычно используется конструкция ***select case*** для определения того, какое сообщение получила оконная процедура и как его обрабатывать. Если оконная процедура обрабатывает сообщение, то ее возвращаемым значением должен быть 0. Все сообщения, не обрабатываемые оконной процедурой, должны передаваться функции Windows, которая называется ***DefWindowProc***. Значение, возвращаемое этой функцией, должно быть возвращаемым значением оконной процедуры.

Первое сообщение, которое получает оконная процедура – это *WМ\_СRЕАТЕ*. Когда Windows обрабатывает функцию ***СreateWindowEx*** в ***WinMain***, она вызывает ***MainWndProc*** с дескриптором окна в качестве первого параметра и с *WМ\_СRЕАТЕ* в качестве второго. ***MainWndProc*** обрабатывает сообщение *WМ\_СRЕАТЕ* и передает управление обратно в Windows, которая, в свою очередь, возвращается обратно в ***WinMain***.

Следующим обрабатывается сообщение *WM\_SIZE*. Определяются размеры рабочей области, панели инструментов и панели состояния. Windows посылает в оконную процедуру сообщение *WM\_SIZE* при любом изменении размеров окна. Переменная *lParam*, переданная в оконную процедуру, содержит ширину рабочей области в младшем слове и высоту в старшем слове.

Сообщения *WM\_PARENTNOTIFY* и *WM\_RBUTTONDOWN* обрабатывают сообщения, поступающие соответственно от панели инструментов и всплывающего меню.

В программе необходимо обрабатывать символы клавиатуры, поэтому она должна обрабатывать сообщения *WM\_CHAR*. Параметр *lParam*, передаваемый в оконную процедуру как часть символьного сообщения, является таким же, как параметр *lParam* аппаратного сообщения клавиатуры, из которого сгенерировано символьное сообщение. Параметр *wParam* – это код символа ASCII. Особым образом обрабатываются клавиши <Backspace>, <Tab> и <Enter>. Все остальные символы обрабатываются похожим образом.

Сообщение *WM\_KEYDOWN* используется для определения нажатой клавиши или комбинации клавиш.

При щелчке мыши на разрешенном пункте меню генерируется сообщение *WM\_COMMAND* (при выборе недоступного или запрещенного пункта меню данное сообщение не генерируется).

Сообщение *WМ\_РАINТ* сообщает программе, что часть или вся рабочая область окна недействительна, и ее следует перерисовать.

При создании окна недействительна вся рабочая зона, поскольку программа еще ничего в окне не нарисовала. При изменении размера окна рабочая область также становится недействительной и содержимое рабочей области не сохраняется. В графической среде это привело бы к тому, что пришлось бы хранить слишком много данных. Вместо этого Windows делает недействительным все окно. Оконная процедура получает сообщение *WМ\_РАINТ* и восстанавливает содержимое окна.

При перемещении окон так, что они перекрываются, Windows не сохраняет ту часть окна, которая закрывается другим окном. Когда эта часть окна позже открывается, Windows помечает его как недействительное. Оконная процедура получает сообщение *WМ\_РАINТ* для восстановления содержимого окна.

Последним сообщением является сообщение *WМ\_DESTROY*. Это сообщение показывает, что Windows находится в процессе ликвидации окна в ответ на полученную от пользователя команду. Программа стандартно реагирует на это сообщение, вызывая ***PostQuitMessage*** (0). Эта функция помещает сообщение *WМ\_QUIT* в очередь сообщений программы. Когда функция ***GetMessage*** получает это сообщение, работа цикла обработки сообщений прерывается и программа заканчивает свою работу, выходя в систему.

**3.3 Модальный и немодальный диалог**

Окно диалога – это окно, которое прикладные программы используют для индицирования или запроса информации. Наиболее часто окна диалога используются для получения от пользователя дополнительной информации сверх той, которую может обеспечить меню.

Окно диалога обычно имеет вид всплывающего окна с разнообразными дочерними окнами элементов управления внутри. Размер и расположение этих дочерних окон задается в шаблона окна диалога в файле описания ресурсов программы. Windows 95 обеспечивает возможность создания всплывающих окон диалога и дочерних окон элементов управления в нем, а также возможность обработки оконной процедурой сообщений окна диалога. При использовании дочерних окон элементов управления, Windows берет на себя решение многих задач (например, управляет всей логикой переключения фокуса ввода между дочерними окнами элементов управления в окне диалога).

Оконная процедура окна диалога – это функция многократного вызова, которую вызывает Windows, когда она имеет сообщение для окна диалога. Как правило, внутри процедуры диалога не реализуется слишком много функций. Исключение составляют лишь инициализация дочерних окон элементов управления при создании окна диалога, обработка сообщений от дочерних окон элементов управления и завершение работы с окном диалога (т.е. обрабатываются сообщения *WM\_INITDIALOG*, *WM\_COMMAND, WM\_DESTROY*). Оконная процедура окна диалога возвращает FALSE (0), если она не обрабатывает сообщение, или TRUE (ненулевое значение), если обрабатывает.

Рассмотрим отдельно модальные и немодальные окна диалога.

Модальное окно диалога.

Модальное окно диалога – это окно, которое индицирует информацию и запрос на ввод от пользователя. Оно называется модальным, поскольку делает родительское окно временно недоступным и заставляет пользователя закончить запрашиваемые действия перед возвратом управления родительскому окну, т.е. нельзя переключаться между окном диалога и другими окнами программы. Но, несмотря на наличие на экране окна диалога, пользователь может переключиться на другие программы. Некоторые окна диалога (называемые системными модальными окнами – *system modal*) этого делать не позволяют. Системное модальное окно диалога вынуждает пользователя, перед тем как он получит возможность сделать что-либо другое в Windows, завершить работу с ним.

Модальное окно диалога запускает свой собственный цикл обработки сообщений из очереди прикладной программы без возврата в функцию ***WinMain***.

Для создания модального окна диалога программа вызывает функцию ***DialogBox*** (могут также использоваться функции ***DialogBoxIndirect****,* ***DialogBoxParam****,* ***DialogBoxIndirectParam***):

*nResult =* ***DialogBox*** *(hInstance, lpTemplate, hWndParent, lpDialogFunc).*

*nResult* – переменная, принимающая значение -1 при неуспешном завершении функции, в обратном случае возвращаемое значение – параметр *nResult* функции ***EndDialog***, которая завершает работу модального окна диалога;

*hInstance* – дескриптор экземпляра;

*lpTemplate* – этот параметр идентифицирует шаблон окна диалога;

*hWndParent* – дескриптор родительского окна модального окна диалога;

*lpDialogFunc* – адрес процедуры диалога.

Работа модального окна диалога завершается с помощью функции ***EndDialog***:

*bret =* ***EndDialog*** *(hDlg, nResult).*

*nResult –* определяет значение, которое будет возвращено приложению от функции ***DialogBox***, создавшей диалоговое окно;

*hDlg* – дескриптор диалогового окна.

Логическая переменная bret принимает значение TRUE или FALSE соответственно при успешном или неуспешном завершении.

Windows не возвращает управление родительскому окну до тех пор, пока не вызвана функция ***EndDialog***.

Немодальное окно диалога.

В отличие от модального окна диалога, немодальное окно диалога не делает недоступным родительское окно, позволяя тем самым пользователю переключаться между окном диалога и окном, в котором оно было создано, а также между окном диалога и остальными программами.

Немодальное окно диалога получает информацию через цикл обработки сообщений функции **WinMain** (см. п. 3.1).

Для создания немодального окна используется функция ***CreateDialog*** (могут также использоваться функции ***CreateDialogIndirect****,* ***CreateDialogParam****,* ***CreateDialogIndirectParam****):*

*hDlgModeLess =* ***CreateDialog*** *(hInstance, lpTemplate, &*

*hWndParent,* *lpDialogFunc)*.

Параметры этой функции те же, что и параметры функции ***DialogBox***. Отличие состоит в том, что дескриптор окна диалога возвращается сразу же после выполнения функции ***CreateDialog*** и, как правило, хранится в глобальной переменной.

Для закрытия немодального окна диалога используется функция ***DestroyWindow***:

*bret =* ***DestroyWindow*** *(hDlg).*

*hDlg* – дескриптор диалогового окна (локальная переменная, используемая в оконной процедуре окна диалога).

Функция возвращает значение bret, равное TRUE или FALSE соответственно при успешном или неуспешном завершении. После вызова функции ***DestroyWindow*** глобальная переменная, содержащая дескриптор немодального окна диалога, должна быть установлена в ноль.

К немодальным окнам диалога относятся окна панели состояния и панели инструментов, используемые в КП Компоновщик. Эти два окна всегда находятся на экране.

**3.4 Панель инструментов**

Панель инструментов предназначена для реализации дополнительных функций, предоставляемых программой.

Панель инструментов представляет собой немодальное окно диалога, содержащее набор битовых картинок и стандартные (зарегистрированные) дочерние окна управления. Выводится панель инструментов в верхнюю часть главного (родительского) окна, размер и местоположение панели устанавливаются с помощью функции ***SetWindowPos***.

Вид панели инструментов представлен на рис. 3.2.



Рис. 3.2. Вид панели инструментов

В рассматриваемой программе на панель инструментов вынесены следующие наиболее часто употребимые функции:

◊ вставить копию текущей строки;

◊ вставить новую строку после текущей;

◊ удалить текущую строку;

◊ восстановить ранее удаленные строки;

◊ ввести шаблон поиска (эта функция реализована с помощью стандартного дочернего окна управления класса «окно редактирования»);

◊ поиск по шаблону вниз;

◊ поиск по шаблону вверх;

◊ сортировать в порядке возрастания;

◊ сортировать в порядке убывания;

◊ буфер обмена;

◊ калькулятор;

◊ печать;

◊ выбор титров;

◊ локальная помощь;

◊ помощь.

Кроме того, на панели инструментов имеются радиокнопки, предназначенные для реализации дополнительных функций подготовки и обработки данных: трансляции, протокола, редактирования названий узлов и др.

Каждый раз при открытии дочернего окна панель инструментов активизирует доступные ему функции.

Входные параметры панели инструментов.

В рассматриваемой программе панель инструментов создается путем вызова функции ***CreateDialog***:

*hPnUpr =* ***CreateDialog*** *(hInstance, LOC («PnUpr» C), hWnd, LOC(PnUpr)).*

Параметры этой функции содержат следующую информацию:

*hInstance* – дескриптор экземпляра программы;

*LOC(‘’PnUpr’’C)* – идентифицирует шаблон блока диалога. Этот параметр является адресом символьной строки, в которой содержится имя шаблона диалогового окна;

*hWnd* – дескриптор родительского окна. Для панели инструментов родительским является главное окно программы;

*LOC(PnUpr)* – адрес процедуры диалога.

Функция возвращает дескриптор панели инструментов, который хранится в глобальной переменной *hPnUpr*.

Процедура диалогового окна панели инструментов определяется следующим образом:

*integer function* ***PnUpr*** *(hDlg, message, wParam, lParam)*

Параметры процедуры диалога аналогичны параметрам оконной процедуры главного окна: *hDlg* – дескриптор окна диалога; *message* – переменная, идентифицирующая сообщение; параметры *wParam* и *lParam* содержат дополнительную информацию о сообщении.

Процедура диалогового окна панели инструментов обрабатывает только два типа сообщений:

а) *WM\_INITDIALOG*; это сообщение посылается процедуре диалога при создании диалогового окна прежде, чем оно отображается на дисплее. При получении этого сообщения процедура диалога вычисляет высоту панели инструментов, используя для этого функцию ***GetWindowRect***:

*bret =* ***GetWindowRect*** *(hDlg, rect).*

Функция заполняет поля структуры *rect* типа *T\_RECT* информацией о размерах рабочей области диалогового окна панели инструментов. Затем вычисляется высота диалогового окна:

*YDlg = rect % bottom – rect % top.*

Вычисленное значение хранится в глобальной переменной *YDlg*.

б) дочерние окна управления панели инструментов посылают процедуре диалогового окна сообщения *WM\_COMMAND*. Параметры этого сообщения содержат идентификатор дочернего окна (младшее слово параметра *wParam*), код уведомления (старшее слово *wParam*) и дескриптор дочернего окна управления, пославшего сообщение (*lParam*). В рассматриваемой программе дочерними окнами управления панели инструментов являются окно ввода шаблона поиска и радиокнопки. Процедура диалога при получении сообщения *WM\_COMMAND* определяет по младшему слову параметра *wParam* идентификатор дочернего окна, от которого пришло сообщение, и выполняет соответствующие действия.

Выходные параметры панели инструментов.

Выходными параметрами панели инструментов являются сообщения активному на данный момент дочернему окну или панели состояния.

Сообщение активному дочернему окну посылается в случае, если левой кнопкой мыши выбран какой-либо орган управления на панели инструментов. Щелчок правой кнопкой мыши на любом органе управления (исключая окно ввода шаблона поиска) заставляет программу вывести в панель состояния текст, поясняющий предназначение выбранного органа управления.

Алгоритм, реализующий вышесказанное, выглядит следующим образом.

1. Нажата кнопка мыши?

Если нажата правая кнопка:

а) определить выбранный на панели инструментов орган управления;

б) вывести соответствующее выбранному органу управления текстовое пояснение в предназначенное для этого поле панели состояния.

Если дочернее окно активно и нажата левая кнопка:

а) определить выбранный на панели инструментов орган управления;

б) отправить активному дочернему окну сообщение, содержащее идентификатор выбранного органа управления.

2. Продолжить выполнение программы.

Приведенный алгоритм требует некоторых пояснений.

Каждый раз при щелчке правой или левой кнопкой мыши над рабочей областью панели инструментов оконной процедуре главного окна посылается сообщение *WM\_PARENTNOTIFY*, содержащее в своих параметрах следующую информацию:

◊ идентификатор панели инструментов;

◊ какая именно кнопка мыши нажата;

◊ координаты курсора в момент нажатия.

Оконная процедура при обработке этого сообщения вызывает последовательно две подпрограммы – ***HelpPnUpr*** и (при условии, что дочернее окно активно) ***GetIdPnUpr***, в которых и реализуется первый шаг алгоритма.

Когда дочернему окну посылается сообщение на поиск по шаблону, оно, в свою очередь, посылает сообщение *IDOK* панели инструментов для насильственного чтения шаблона.

По окончании работы подпрограмм возобновляется выполнение цикла обработки сообщений.

В рассматриваемой программе процедура диалога панели инструментов и указанные подпрограммы объединены в одном модуле, что позволяет рассматривать панель инструментов как «черный ящик» (объект). «Черный ящик» получает некоторые сообщения и определенным образом на них реагирует (рис. 3.3).

*WM\_INITDIALOG*

*WM\_PARENTNOTIFY*

**Панель**

**инструментов**

*WM\_COMMAND*

Инициализация

Сообщение активному дочернему окну или панели состояния

*IDOK*

Рис. 3.3. Панель инструментов

Панель инструментов отображается на поверхности главного окна все время и уничтожается только по завершении работы.

**3.5 Панель состояния (строка состояния)**

Панель состояния предназначена для вывода на экран необходимой пользователю информации.

Панель состояния представляет собой немодальное окно диалога, которое выводится в нижнюю часть главного (родительского) окна. Размер и местоположение панели устанавливаются с помощью функции ***SetWindowPos***.

Вид панели состояния показан на рис. 3.4.



Рис. 3.4. Вид панели состояния.

В рассматриваемой программе панель состояния используется для отображения краткого описания органов управления панели инструментов при их просмотре пользователем, а при наличии активного дочернего окна – и для отображения количества записей и номера записи, с которой пользователь работает в текущий момент времени. Поля панели состояния, в которых отображается вышеуказанная информация, представляют собой статические дочерние окна управления (предопределенного класса «статическое окно»). Панель состояния содержит также один орган управления общего пользования – т.н. прогресс-индикатор, отображающий динамику длительной операции как процент от выполненной задачи.

Входные параметры панели состояния.

В рассматриваемой программе панель состояния создается функцией ***CreateDialog***:

*hPnSost =* ***CreateDialog*** *(hInstance, LOC («PnSost» C), hWnd, LOC(Pnsost)).*

Параметры этой функции являются входными параметрами для панели состояния. Первый и третий из них аналогичны соответствующим входным параметрам панели инструментов. Параметр *LOC («PnSost» C)* идентифицирует шаблон диалогового окна панели состояния. Параметр *LOC(Pnsost)* содержит адрес процедуры диалога панели состояния.

Функция возвращает дескриптор панели состояния, который хранится в глобальной переменной *hPnSost*.

Процедура диалогового окна панели состояния определяется следующим образом:

*integer function* ***Pnsost*** *(hDlg, message, wParam, lParam).*

Ее параметры аналогичны параметрам процедуры диалога панели инструментов.

В рассматриваемой программе процедура диалогового окна панели состояния получает и обрабатывает только сообщение *WM\_INITDIALOG*. При получении этого сообщения процедура диалога вычисляет высоту панели состояния аналогично тому, как это реализовано в процедуре диалога панели инструментов. Вычисленное значение хранится в глобальной переменной *YPnSost*.

Отображение информации.

Для вывода информации в дочернее окно управления панели состояния используется функция

***SetDlgItemText*** *(hwndDlg***,** *idControl***,** *lpsz).*

Параметры этой функции следующие:

*hwndDlg* – дескриптор диалогового окна (в нашем случае это глобальная переменная *hPnSost*);

*idControl* – идентификатор дочернего окна управления;

*lpsz* – указатель на Си-строку, содержащую текст, который нужно вывести в данное дочернее окно управления.

Для управления прогресс-индикатором ему посылаются сообщения при помощи функции

**SendDlgItemMessage (***hwndDlg***,** *idControl***,** *Msg***,** *wParam***,** *lParam).*

Параметр Msg здесь – переменная, содержащая посылаемое сообщение, а параметры *wParam* и *lParam* содержат дополнительную информацию о сообщении.

Панель состояния отображается на поверхности главного окна все время работы программы и уничтожается только по завершении работы.

**3.6 Строковый редактор**

Строковый редактор предназначен для отображения и редактирования текстовых данных.

Строковый редактор является дочерним окном оконного редактора и выводится на его рабочую поверхность. Размер и местоположение строкового редактора устанавливаются с помощью функции ***SetWindowPos*** в оконном редакторе. Высота окна строкового редактора определяется высотой используемого шрифта, ширина – длиной редактируемого текста.

Работа со строковым редактором.

Оконный редактор содержит текстовые данные в виде страницы, состоящей из нескольких строк. Каждая строка разбивается на поля. Каждое поле – это определенное число (параметр). Содержимое текущего поля, т.е. поля, доступного для редактирования, помещается в окно строкового редактора (далее – «окно»), которое отмечено двойным бордюром.

Каретка перемещается внутри «окна». Если каретка достигает границы «окна» (левой или правой), то дальнейшее движение в том же направлении приводит к перемещению «окна». Если же «окно» находится на левой или правой границе страницы, то его дальнейшее перемещение в том же направлении приводит к «листанию» страницы вправо / влево.

Перемещения «окна» по странице (страницам) возможны следующие: вверх / вниз на одну строку; влево / вправо на одно поле; в начало / конец строки; на ту же позицию предыдущей / последующей страницы («листание» вверх / вниз); на ту же позицию первой / последней страницы.

Ввод текстовых данных и все коррекции производятся внутри «окна». Если каретка находится на правой границе «окна», то после ввода «окно» перемещается вправо на одно поле (если это возможно).

Внутри «окна» символы можно удалять по одному или группами (слева / справа от позиции каретки, удалить все содержимое «окна»).

Создание окна строкового редактора.

Создание окна строкового редактора происходит при обработке сообщения *WM\_CREATE* в оконной процедуре оконного редактора. Для этого используется рассмотренная ранее функция ***CreateWindowEx***. Она возвращает дескриптор окна строкового редактора, который хранится в глобальной переменной *hWEdit*. Следует отметить, что при регистрации класса окна строкового редактора стиль окна задается как комбинация следующих стилей:

◊ *CS\_VREDRAW*, *CS\_HREDRAW* – разрешают перерисовку рабочей области окна в случае изменения его высоты или ширины;

◊ *CS\_OWNDC* – создает уникальный контекст устройства для каждого окна, созданного на основе данного класса;

◊ *CS\_DBLCLKS* – разрешает посылку оконной процедуре сообщения о двойном щелчке мышью над рабочей областью окна.

При создании окна строкового редактора используется «расширенный» стиль *WS\_EX\_DLGMODALFRAME*, создающий двойную рамку вокруг окна.

Оконная процедура строкового редактора определяется следующим образом:

*integer function* ***Editw*** *(hEdit, mesg, wParam, lParam)*

Ее параметры аналогичны параметрам оконной процедуры главного окна: *hEdit* – дескриптор окна строкового редактора; *mesg* – переменная, идентифицирующая сообщение; параметры *wParam* и *lParam* содержат дополнительную информацию о сообщении.

Вид строкового редактора представлен на рис. 3.5.



Рис. 3.5. Вид строкового редактора

Входные параметры строкового редактора.

Входными параметрами для строкового редактора являются нажатия алфавитно-цифровых и некоторых функциональных клавиш. Он также отслеживает состояние системных клавиш для задействования т.н. «горячих клавиш». Строковый редактор должен реагировать и на нажатия левой или правой кнопок мыши.

Выходные параметры строкового редактора.

Выходными параметрами строкового редактора являются:

◊ отображение редактируемого текста;

◊ сообщение *WM\_LBUTTONDBLCLK*, посылаемое оконной процедуре родительского окна;

◊ сообщение *WM\_COMMAND*, посылаемое оконной процедуре родительского окна.

Сообщение *WM\_COMMAND* посылается в случае, если необходимо переместить окно строкового редактора или нажата «горячая клавиша».

Алгоритм работы строкового редактора.

1. Отображение редактируемого текста.
2. Обработка сообщений, связанных с вводом с клавиатуры.
3. Обработка сообщений, связанных с нажатиями кнопок мыши.

При работе по приведенному алгоритму используются следующие переменные:

◊ *Bufed* – символьная строка, содержащая редактируемый текст;

◊ *Leng* – длина редактируемого текста (содержимого Bufed) в символах;

◊ *Poz* – положение каретки (число символов) от левого края окна;

◊ *Cod* – главный код нажатой клавиши;

◊ *Sccod* – информационный код нажатой клавиши;

◊ *Codv* – код возврата (0‑нет коррекций; 1‑есть коррекция).

Все переменные, кроме *Bufed*, – целочисленные.

Первый пункт алгоритма реализуется следующим образом.

После создания окна строкового редактора, но перед его отображением, оконная процедура получает сообщение *WM\_CREATE*. При его обработке обнуляются переменные *Cod*, *Sccod*, *Codv*. Задаются шрифт и цвет выводимого в окно текста. Окно получает фокус ввода. Вывод редактируемого текста, т.е. содержимого *Bufed*, происходит при получении сообщения *WM\_PAINT* и осуществляется функцией ***TextOut***:

*bret =* ***TextOut (****hdc****,*** *nXStart****,*** *nYStart****,*** *lpString****,*** *cbString),*

где: *bret* – логическая переменная, принимающая значение TRUE или FALSE соответственно при успешном или неуспешном завершении функции; *hdc* – дескриптор контекста устройства, на которое выводится текст; *nXStart* и *nYStart* – координаты начальной позиции вывода текста; *lpString* – указатель на символьную строку, которая содержит текст; *cbString* – количество символов в этой строке.

Затем каретка устанавливается на позицию, определяемую переменной *Poz* (если каретка отсутствует, она предварительно создается и отображается).

При получении сообщения *WM\_DESTROY* обнуляется переменная *hWEdit*, уничтожается каретка и закрывается окно строкового редактора.

Второй пункт алгоритма.

Для удобства работы в Компоновщике расчетных схем предусмотрена возможность использования т.н. «горячих клавиш». «Горячая клавиша» – это комбинация клавиш, которую пользователь может нажать для вызова к.-л. функции. В рассматриваемой программе используются две группы комбинаций: CTRL + <буква>, ALT + <буква>. Отслеживание к.-л. комбинации из первой группы не вызывает особых затруднений, в то время как использование клавиши ALT имеет свои особенности.

ОС Windows нажатия клавиш делятся на системные и несистемные. Системные нажатия – это нажатия клавиш, используемых для управления приложением и Windows. В IBM PC для этой цели Windows использует клавишу ALT, поэтому можно сказать, что нажатия клавиши ALT являются системными. При нажатии клавиши ALT (или любой другой клавиши, если ALT при этом удерживается нажатой) оконная процедура имеющего фокус ввода окна получает сообщение *WM\_SYSKEYDOWN*. Нажатия всех остальных клавиш считаются несистемными, в этом случае оконная процедура окна с фокусом ввода получает сообщение *WM\_KEYDOWN*.

В оконной процедуре строкового редактора при получении любого из этих сообщений фиксируется информационный код нажатой клавиши (*sccod*), переменная *cod* обнуляется. Затем вызывается подпрограмма ***edit*** с параметром *hEdit*.

При получении сообщения *WM\_CHAR*, генерируемого рассмотренной ранее функцией ***TranslateMessage***, фиксируется главный код нажатой клавиши (*cod*), переменная *sccod* обнуляется, а затем также вызывается подпрограмма ***edit*** с параметром *hEdit*.

Предусмотрена перекодировка из Windows‑кодировки в MS DOS‑кодировку. При необходимости этот модуль можно убрать.

В подпрограмме edit производятся следующие операции:

а) если нажата одна из тех «горячих клавиш», которые дублируют органы управления панели инструментов, то параметру *wParam* присваивается значение идентификатора соответствующего органа управления и в оконную процедуру родительского окна посылается сообщение *WM\_COMMAND*.

б) если нажата одна из тех «горячих клавиш», которые дублируют операции выделения строк или блоков, операцию снятия маркировки, то параметру *wParam* присваивается значение идентификатора соответствующего пункта всплывающего меню оконного редактора и в его оконную процедуру посылается сообщение *WM\_COMMAND*.

в) если нажата клавиша «ESCAPE» или «ENTER», переменной *sccod* присваивается соответственно 0 или значение, соответствующее информационному коду клавиши «стрелка вправо», а затем оконной процедуре оконного редактора посылается сообщение *WM\_COMMAND* с параметром *wParam*, равным -1.

г) если значение переменной *cod* не равно нулю, т.е. была нажата клавиша в алфавитно-цифровой части клавиатуры, происходит изменение редактируемого текста, переменной *sccod* присваивается значение, соответствующее информационному коду клавиши «стрелка вправо».

д) если значение переменной *cod* равно нулю, по информационному коду нажатой клавиши проверяется ее принадлежность к группе функциональных клавиш или клавиш управления курсором, которые выполняют следующие действия:

◊ клавиша «стрелка влево» (*VK\_LEFT*) – каретка перемещается на одну позиции влево. Если одновременно с клавишей была нажата клавиша Ctrl или каретка вышла за левую границу окна строкового редактора, оконной процедуре оконного редактора посылается сообщение *WM\_COMMAND* с параметром *wParam*, равным -1;

◊ клавиша «стрелка вправо» (*VK\_RIGHT*) – каретка перемещается на одну позиции вправо. Если одновременно с клавишей была нажата клавиша Ctrl или каретка вышла за правую границу окна строкового редактора, оконной процедуре оконного редактора посылается сообщение *WM\_COMMAND* с параметром *wParam*, равным -1;

◊ клавиша «Delete» (*VK\_DELETE*) – происходит удаление символа над кареткой, переменной *codv* присваивается значение 1;

◊ клавиша «Home» (*VK\_HOME*) – происходит перемещение каретки в начало окна. Если одновременно с клавишей была нажата клавиша Ctrl, оконной процедуре оконного редактора посылается сообщение *WM\_COMMAND* с параметром *wParam*, равным -1;

◊ клавиша «End» (*VK\_END*) – происходит перемещение каретки в конец окна. Если одновременно с клавишей была нажата клавиша Ctrl, оконной процедуре оконного редактора посылается сообщение *WM\_COMMAND* с параметром *wParam*, равным -1;

◊ клавиша «F5» (*VK\_F5*) – происходит удаление текста слева от каретки;

◊ клавиша «F6» (*VK\_F6*) – происходит удаление текста справа от каретки;

◊ клавиша «F7» (*VK\_F7*) – происходит удаление всего текста.

При изменении редактируемого текста код возврата принимает значение, равное 1, а оконная процедура строкового редактора получает сообщение *WM\_PAINT* путем вызова функции ***InvalidateRect***.

Третий пункт алгоритма.

При получении оконной процедурой строкового редактора сообщения *WM\_LBUTTONDOWN* определяются координаты курсора мыши и каретка перемещается в позицию (с учетом ширины символов), соответствующую местоположению курсора мыши.

При получении оконной процедурой строкового редактора сообщения *WM\_RBUTTONDOWN* на экран выводится всплывающее меню, вид которого представлен на рис. 3.6.

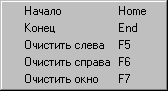


Рис. 3.6. Всплывающее меню строкового редактора

При щелчке левой кнопкой мыши на к.-л. пункте всплывающего меню оконной процедуре строкового редактора посылается сообщение *WM\_COMMAND*, содержащее в младшем слове параметра *wParam* идентификатор выбранного пункта меню. Переменной *sccod* присваивается значение идентификатора выбранного пункта меню и вызывается подпрограмма ***edit.***

Двойной щелчок левой кнопкой мыши на рабочей области окна строкового редактора порождает сообщение *WM\_LBUTTONDBLCLK*. При получении этого сообщения оконная процедура строкового редактора «переправляет» его родительскому окну, т.е. оконному редактору, где оно и обрабатывается.

В рассматриваемой программе оконная процедура строкового редактора и подпрограмма ***edit*** объединены в одном модуле, что позволяет рассматривать строковый редактор как «черный ящик». «Черный ящик» получает некоторые сообщения и определенным образом на них реагирует (рис. 3.7).

**Строковый редактор.**

Переменные:

*hWEdit,*

*Bufed, Leng,* *Poz*,

*Cod*, *Sccod, Codv*

*WM\_KEYDOWN*

*WM\_SYSKEYDOWN*

*WM\_CHAR*

*WM\_LBUTTONDOWN*

*WM\_ LBUTTONDBLCLK*

*WM\_RBUTTONDOWN*

Отображение и

редактирование текста

Сообщение

*WM\_ LBUTTONDBLCLK* оконному редактору

Сообщение

*WM\_ COMMAND*

оконному редактору

Рис. 3.7. Строковый редактор

Строковый редактор отображается на поверхности оконного редактора все время и уничтожается только по завершении его работы.

**3.7 Оконный редактор**

Оконный редактор предназначен для отображения и редактирования базы данных.

Оконный редактор является дочерним окном главного окна и выводится на его рабочую поверхность. Размер и местоположение оконного редактора устанавливаются с помощью функции ***SetWindowPos***. Ширина окна определяется длиной строки базы данных, а высота – условием неперекрытия диалоговых окон (панели инструментов, панели состояния и, если активен режим ввода названий узлов, окна ввода названий узлов).

Создание оконного редактора.

Для создания окна оконного редактора вызывается функция ***CreateWch***, которая определяется следующим образом:

*hWch =* ***CreateWch*** *(hWnd, szClassName, UnschWProc, LengStr),*

где: *hWnd* – дескриптор родительского окна (т.е. главного окна); *szClassName* – имя класса окна (оно же является и заголовком окна); *UnschWProc* – имя процедуры обработки сообщений оконного редактора; *LengStr* – длина строки базы данных.

В процессе выполнения этой функции вычисляется максимальная ширина окна в пикселях, регистрируется класс окна (причем устанавливается предопределенный в Windows стиль курсора «*IDC\_CROSS*», т.е. «перекрестие»). Затем при помощи функции ***CreateWindowEx*** создается окно оконного редактора. Стиль окна – дочернее, видимое, имеет строку заголовка с кнопкой закрытия окна, вертикальную и горизонтальную полосы прокрутки.

Функция ***CreateWch*** возвращает дескриптор оконного редактора, который хранится в глобальной переменной *hWch*.

Оконная процедура оконного редактора определяется следующим образом:

*integer function* ***UnschWProc*** *(hWch, mesg, wParam, lParam)*

Ее параметры аналогичны параметрам оконной процедуры главного окна: *hWch* – дескриптор оконного редактора; *mesg* – переменная, идентифицирующая сообщение; параметры *wParam* и *lParam* содержат дополнительную информацию о сообщении.

Вид оконного редактора представлен на рис. 3.8 (см. на следующей странице).

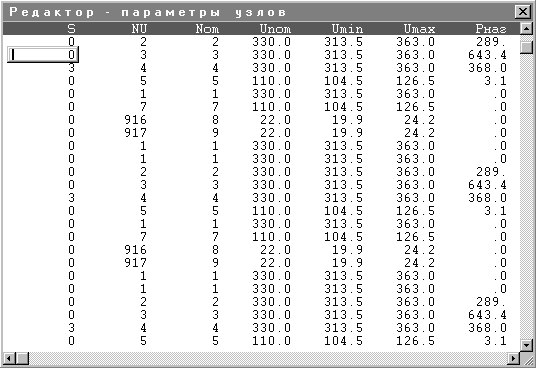


Рис. 3.8. Вид оконного редактора

Входные параметры оконного редактора.

Входными параметрами для оконного редактора являются сообщения о нажатии кнопок мыши, перемещении мыши (сообщение *WM\_MOUSEMOVE*), сообщения от полос прокруток (*WM\_VSCROLL* и *WM\_HSCROLL*). Как говорилось ранее, оконный редактор может получать сообщение *WM\_COMMAND* от панели инструментов и строкового редактора, а также сообщение *WM\_LBUTTONDBLCLK*.

Выходные параметры оконного редактора.

Выходными параметрами оконного редактора являются:

◊ отображение редактируемой базы данных и строкового редактора;

◊ реализация дополнительных функций, предоставляемых рассматриваемой программой.

Примечание: вызов дополнительных функций может осуществляться через «горячие клавиши», через панель инструментов или через всплывающее меню оконного редактора.

Алгоритм работы оконного редактора.

1. Отображение редактируемой базы данных и строкового редактора.
2. Обработка сообщений мыши.
3. Обработка сообщений полос прокрутки.
4. Обработка сообщения *WM\_COMMAND*.
5. Реализация дополнительных функций, предоставляемых программой.

Смысл некоторых переменных, использующихся при работе по приведенному алгоритму, поясняется на рис. 3.9.

**Строка титров**

*jsMax*

*Kol*

*MaxCol*

*Row*

*MaxRow*

*in*

*iv*

*jsLeft*

*js*

*jsRight*

***База данных***

***Оконный редактор***

- строковый редактор

Рис. 3.9

*js, jsMax* – номер текущей записи и максимальное число полей в записи базы данных соответственно;

*jsLeft, jsRight* – границы отображаемого текста базы данных (содержимого оконного редактора) по горизонтали (номера столбцов);

*MaxRow, MaxCol* – максимальное число символов, отображаемых в оконном редакторе, по вертикали и горизонтали соответственно;

*in, iv* – границы отображаемого текста базы данных (содержимого оконного редактора) по вертикали (номера записей);

*Row –* номер записи, на которой находится строковый редактор (принимает значения от 1 до *MaxRow*);

*Kol* – количество записей (строк) в базе данных.

Используются также следующие переменные и массивы:

*Text* – исходный массив (база данных);

*Bufv* – символьный массив, отображаемый на экране (содержимое оконного редактора);

*Par* – целочисленный массив, содержащий границы полей базы данных (табуляция);

*Ndn, Nup* – минимальный и максимальный номера строк массива *Bufv*, в пределах которых его необходимо заполнить новыми данными из *Text*;

*BVpol, Bhpol* – логические переменные, указывающие на наличие полос прокрутки.

Все упомянутые переменные, кроме *BVpol* и *Bhpol* – целочисленные. Назначение некоторых других переменных, используемых в программе, будет пояснено далее.

Первый пункт алгоритма реализуется следующим образом.

После создания оконного редактора, но перед его отображением, оконная процедура получает сообщение *WM\_CREATE*. При его обработке путем вызова функции ***SetStatePnUpr*** на панели инструментов активизируются доступные оконному редактору функции. Затем вызывается функция ***WMCREATE***:

*Ret =* ***WMCREATE*** *(hWch, NumbRowTitr, Par),*

где *NumbRowTitr* – число строк титров (в рассматриваемой программе равно 1).

В процессе выполнения этой функции задаются шрифт и цвет выводимого в окно текста, рассчитываются значения переменных *js, jsMax,* *jsLeft, jsRight, MaxRow, MaxCol.* После этого вызывается подпрограмма коррекции данных ***corUnsch*** с параметром *hWch*, в которой данные подготавливаются для вывода на экран (более подробно работа подпрограммы ***corUnsch*** будет рассмотрена далее). Затем путем вызова функции ***CreateEdit*** с параметром *hWch* создается строковый редактор.

При получении сообщения *WM\_PAINT* оконная процедура вызывает функцию ***WMPAINT***:

*ret =* ***WMPAINT*** *(hWch, TitrUnsch, NumbRowTitr, Par, Bufv),*

где *TitrUnsch* – символьный массив, содержащий титры.

В процессе выполнения этой функции производится вывод на экран строки (строк) титров и содержимого массива *Bufv*. Если имеются отмеченные строки или блоки текста, они выводятся инвертированными.

При получении сообщения *WM\_SIZE* оконная процедура вызывает функцию ***WMSIZE***:

*ret =* ***WMSIZE*** *(hWch, Kol, NumbRowTitr, LengStr),*

в которой, исходя из нового размера рабочей области оконного редактора, пересчитываются переменные *MaxRow, MaxCol,* и анализируется, нужно ли отображать полосы прокрутки. Далее в оконной процедуре рассчитывается новое значение переменной *jsRight*, а затем, в случае необходимости, корректируется положение окна строкового редактора (при уменьшении размеров рабочей области оконного редактора) или содержимое массива *Bufv* (при увеличении размеров рабочей области оконного редактора).

При получении сообщения *WM\_DESTROY* обнуляется переменная *hWch*, уничтожается окно строкового редактора и закрывается окно оконного редактора.

Рассмотрим более подробно подпрограмму ***corUnsch.***

В случае, если код возврата отличен от нуля, сначала в символьный массив *Bufv*, а затем и в *Text*, заносятся изменения, произведенные в строковом редакторе. Затем вызывается функция ***MoveEditWithMouse:***

*ret =* ***MoveEditWithMouse*** *(Par, Kol),*

Если в результате щелчка левой кнопкой мыши или действия полос прокрутки окно строкового редактора необходимо переместить, функция возвращает положительное значение, в зависимости от которого вызывается функция ***Metka4*** или ***Metka5***, а затем ***Metka6***. Если в результате действия полосы прокрутки требуется изменить содержимое оконного редактора, то пересчитываются границы отображаемого текста базы данных *in, iv,* в соответствии с ними заполняется массив *Bufv*, а затем вызываются функции ***Metka4, Metka5, Metka6****.* Если функция ***MoveEditWithMouse*** возвращает 0, т.е. перемещение окна не производилось ни мышью, ни полосами прокрутки, то вызывается подпрограмма управления окном ***contw***, и в случае перемещения строкового редактора с помощью клавиатуры или всплывающего меню (влево / вправо, вверх / вниз, в начало / конец строки, в начало / конец базы данных, листание вверх / вниз) пересчитывается положение строкового и оконного редакторов. В том случае, если перемещение строкового редактора не производилось, проверяется, был ли запрос на поиск по шаблону.

Функция ***Metka4*** выполняет пересчет переменных *jsLeft, jsRight* в зависимости от того, в каком столбце (*js*) находится строковый редактор. Пересчитывается также его горизонтальная координата *XEdit.*

Функция ***Metka5*** выполняет пересчет вертикальной координаты строкового редактора *YEdit*, а в символьную строку *Bufed* помещается редактируемый текст.

В процессе выполнения функции ***Metka6*** в соответствующие поля панели состояния выводится количество записей в базе данных и номер текущей записи.

Второй пункт алгоритма реализуется следующим образом.

Оконная процедура оконного редактора обрабатывает пять сообщений мыши: *WM\_LBUTTONDOWN, WM\_RBUTTONDOWN, WM\_MOUSEMOVE, WM\_LBUTTONDBLCLK* и *WM\_LBUTTONUP.*

При получении сообщения *WM\_LBUTTONDOWN* вызывается функция ***WMLBUTTONDOWN***:

*ret =* ***WMLBUTTONDOWN*** *(hWch, mesg, lParam, corUnsch).*

В процессе выполнении этой функции переменной *cod* присваивается значение переменной *mesg*, т.е. идентификатор сообщения *WM\_LBUTTONDOWN*, а переменной *sccod* – значение переменной *lParam*, содержащей горизонтальную позицию курсора мыши в младшем слове и вертикальную позицию курсора мыши в старшем слове. Затем вызывается подпрограмма ***corUnsch***, в которой рассчитываются новые координаты окна строкового редактора. Обновляется его содержимое, каретка устанавливается на первую позицию. Для перемещения окна строкового редактора вызывается функция ***SetWindowPos***, после чего управление возвращается в оконную процедуру, где выполняются еще несколько операторов, связанных с реализацией дополнительных функций, и о которых более подробно будет рассказано при описании четвертого пункта алгоритма.

При получении оконной процедурой оконного редактора сообщения *WM\_RBUTTONDOWN* на экран выводится всплывающее меню. Перед его созданием для каждого пункта меню задается идентификатор и отображаемый текст. Затем вызывается подпрограмма ***popupm*** с параметром *hWch*, в которой происходит создание и отображение меню. Вид всплывающего меню оконного редактора представлен на рис. 3.10.

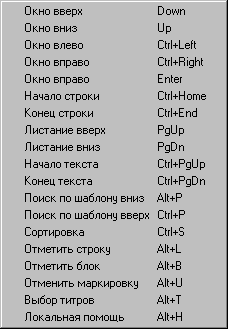


Рис. 3.10. Вид всплывающего меню оконного редактора

При щелчке левой кнопкой мыши на к.-л. пункте всплывающего меню оконной процедуре оконного редактора посылается сообщение *WM\_COMMAND*, содержащее в младшем слове параметра *wParam* идентификатор выбранного пункта меню.

Обработка оконной процедурой сообщений *WM\_MOUSEMOVE, WM\_LBUTTONDBLCLK* и *WM\_LBUTTONUP* необходима для реализации дополнительных функций, предоставляемых программой, и подробнее будет рассмотрена при описании четвертого пункта алгоритма.

Третий пункт алгоритма реализуется следующим образом.

При обработке действий пользователя вертикальная и горизонтальная полосы прокрутки посылают родительскому окну (т.е. оконному редактору) сообщения *WM\_VSCROLL* и *WM\_HSCROLL* соответственно. При этом переменная *wParam* в младшем слове содержит одну из величин, идентифицирующих действие пользователя, и определенных в файле msfwinty.f90 (приведены только те, которые обрабатываются в рассматриваемой программе):

◊ *SB\_LINEDOWN* – прокрутка на строку вниз;

◊ *SB\_LINEUP* – прокрутка на строку вверх;

◊ *SB\_PAGEDOWN* – прокрутка на страницу вниз;

◊ *SB\_PAGEUP* – прокрутка на страницу вверх;

◊ *SB\_THUMBTRACK* – перетаскивание бегунка. Номер текущей позиции бегунка содержится в старшем слове wParam.

При обработке сообщения *WM\_VSCROLL* вызывается функция ***WMVSCROLL***:

*ret =* ***WMVSCROLL*** *(hWch, mesg, wParam, corUnsch).*

При обработке сообщения *WM\_HSCROLL* вызывается функция ***WMHSCROLL***:

*ret =* ***WMHSCROLL*** *(hWch, mesg, wParam, corUnsch).*

Алгоритмы работы этих функций схожи. В случае, если младшее слово *wParam* содержит идентификатор *SB\_LINEDOWN,* *SB\_LINEUP,* *SB\_PAGEDOWN* или *SB\_PAGEUP*, переменная *cod* обнуляется, а переменной *sccod* присваивается значение идентификатора соответствующего пункта всплывающего меню. Если младшее слово *wParam* содержит идентификатор *SB\_THUMBTRACK*, переменной *cod* присваивается значение переменной *mesg*, а переменной *sccod* – значение переменной *wParam*. (Во всех остальных случаях управление возвращается оконной процедуре.) Затем вызывается подпрограмма ***corUnsch***, в которой рассчитываются новые координаты окна строкового редактора. Обновляется его содержимое, каретка устанавливается на первую позицию. Для перемещения окна строкового редактора вызывается функция ***SetWindowPos***, после чего управление возвращается оконной процедуре.

Если указанное сообщение поступило от горизонтальной полосы прокрутки, то соответствующие операции осуществляются влево / вправо.

Четвертый пункт алгоритма.

При получении сообщения *WM\_COMMAND* вспомогательной переменной *ret* присваивается значение младшего слова *wParam*. Дальнейшая работа программы организуется с помощью конструкции *Select Case*, тест-выражением для которой является переменная *ret*.

Если *ret* = -1, т.е. сообщение пришло от оконной процедуры строкового редактора, или *ret* является идентификатором пункта всплывающего меню оконного редактора (кроме последних пяти), вызывается функция ***MYCASE***:

*ret =* ***MYCASE*** *(hWch, ret, corUnsch).*

При выполнении функции сначала анализируется значение переменной *ret*, и если *ret* является идентификатором пункта всплывающего меню, переменная *cod* обнуляется, а *sccod* присваивается значение *ret*. Затем вызывается подпрограмма ***corUnsch***. Обновляется содержимое строкового редактора, каретка устанавливается на первую позицию. Для перемещения окна строкового редактора вызывается функция ***SetWindowPos*** и управление возвращается оконной процедуре.

Если ret является идентификатором радиокнопки или одного из органов управления панели инструментов, т.е. была вызвана одна из функций, доступных данному дочернему окну, выполняется соответствующая переменной *ret Case*‑секция.

Во всех остальных случаях (по оператору *Case Default*) управление передается функции ***DefWindowProc***.

Пятый пункт алгоритма.

В рассматриваемой программе реализованы следующие дополнительные функции:

1. «Выбор титров».
2. «Вызов Протокола».
3. «Ввод названий узлов».
4. «Отметить строку (строки)».
5. «Отметить блок».
6. «Снять маркировку».
7. «Поиск по шаблону вниз / вверх».

Рассмотрим работу каждой из них подробнее.

1. «Выбор титров».

Назначение – предоставление пользователю возможности задавать порядок отображения параметров в любой комбинации и в любой последовательности.

При вызове этой функции на экран выводится окно модального диалога, в котором предлагается выбрать желаемый порядок следования полей записей базы данных. Вид модального диалога представлен на рис. 3.11 (см. на следующей странице).

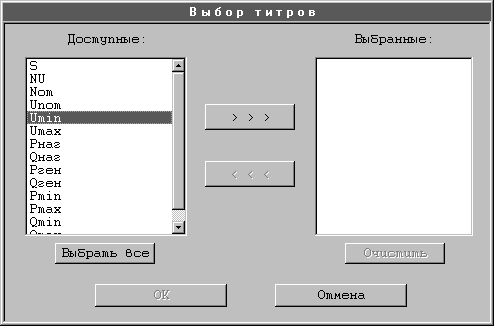


Рис. 3.11. Вид модального диалога «Выбор титров»

Оконная процедура диалога и подпрограммы, необходимые для реализации данной функции, объединены в модуле ***Titr***.

При инициализации диалога оконная процедура вызывает подпрограмму ***TitreInit***, в которой инициализируются необходимые переменные и на экран выводится список титров, доступных для выбора.

В процессе работы с данным диалогом пользователь может выполнять следующие действия.

◊ Нажать кнопку переноса титра вправо. Оконная процедура диалога получает соответствующее сообщение, по которому вызывает подпрограмму ***Move***, где выполняются операции переноса выделенного титра из одного окна списка в другой. Если ни один титр в левом окне списка не выбран или список пуст, кнопка недоступна.

◊ Нажать кнопку переноса титра влево. Вызывается подпрограмма ***Move***, где выполняются операции переноса выделенного титра из одного окна списка в другой. Если ни один титр в правом окне списка не выбран или список пуст, кнопка недоступна.

◊ Нажать кнопку «Выбрать все». Все содержимое левого окна списка добавляется в правый, после чего левый список очищается. Если левый список пуст, кнопка недоступна.

◊ Нажать кнопку «Очистить все». Содержимое обоих окон списков уничтожается, а затем вызывается подпрограмма ***TitreInit***. Данная кнопка недоступна, если левый список пуст.

◊ Нажать кнопку «Отмена». Восстанавливается прежний порядок следования титров, окно диалога завершает свою работу и управление передается оконной процедуре оконного редактора.

◊ Нажать кнопку «Ок». В переменную *jsMax* записывается количество выбранных титров. После этого последовательно вызываются подпрограмма ***Change\_By\_Places\_Titr***, в которой производится коррекция символьного массива *TitrUnsch*, содержащего титры, и подпрограмма ***Change\_By\_Places\_Text***, в которой производится коррекция символьного массива *Bufv* и целочисленного массива табуляции *Par* в соответствии с выбранным порядком отображения параметров. Окно модального диалога уничтожается и управление передается оконной процедуре оконного редактора. Данная кнопка недоступна, пока не будет выбран по крайней мере один титр.

Модуль ***Titr*** также содержит подпрограмму ***Vosvrat\_Titr***, в которой производится восстановление первоначального порядка следования титров, и подпрограмму ***Vosvrat\_Text***, в которой производится восстановление первоначального порядка следования элементов массивов *Bufv* и *Par*.

Глобальная логическая переменная *TitrFlag* по умолчанию устанавливается в FALSE и принимает значение TRUE только в том случае, если порядок отображения параметров был изменен. В процессе работы оконного редактора перед началом обмена данными между массивами *Bufv* и *Text* анализируется состояние переменной *TitrFlag* и, если она установлена в TRUE, вызывается подпрограмма ***Vosvrat\_Text***, а при завершении обмена вызывается подпрограмма ***Change\_By\_Places\_Text***.

2. «Вызов Протокола».

Эта функция становится доступной либо после трансляции любого файла основной базы данных, либо после компоновки расчетной схемы. При ее вызове пользователю предоставляется список обнаруженных при трансляции ошибок. Предусмотрена возможность автоматического поиска источника ошибки, принадлежащего к редактируемому типу исходных данных.

При вызове протокола на экран выводится модальное окно диалога. Вид модального диалога представлен на рис. 3.12.

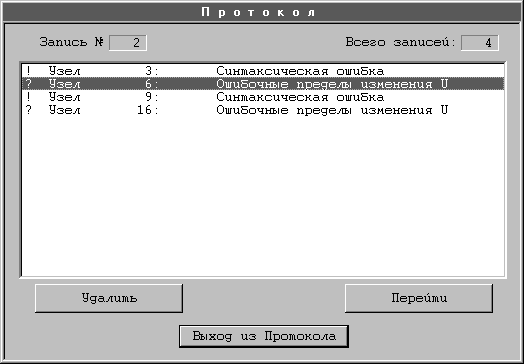


Рис. 3.12. Вид модального диалога «Вызов Протокола»

Оконная процедура диалога и подпрограмма ***ProtocolInit***, которая вызывается при инициализации диалога, объединены в модуле ***ProtMod***.

Каждому сообщению об ошибке соответствует одна строка. Общее количество строк (сообщений) и номер текущей строки указываются в соответствующих полях («Всего записей» и «Запись №»).

Предусмотрены три группы сообщений: «!» – использование данных, как правило, при решении технологических задач приводит к их аварийному завершению; «?» – возможны ошибки при решении ряда технологических задач; «» – предупреждения, не обязательно связанные с ошибками.

Для удаления какого либо сообщения необходимо отметить его с помощью клавиатуры или щелчком левой кнопки мыши, а затем нажать кнопку «Удалить». Удаление является мнимым: все удаленные сообщения легко восстанавливаются – следует закрыть, а затем повторно вызвать протокол.

Автоматический поиск источника ошибки осуществляется следующим образом: сначала необходимо отметить нужное сообщение, а затем нажать кнопку «Перейти». Окно протокола закрывается, в оконный редактор выводится нужная страница данных, а окно строкового редактора устанавливается на записи, содержащей ошибку. Переход к месту ошибки возможен также по двойному щелчку левой кнопкой мыши на сообщении об ошибке.

Выход из окна протокола осуществляется автоматически, если удалены все сообщения об ошибках, или по нажатию кнопки «Выход из Протокола».

При изменении редактируемых данных вызов функции «Протокол» становится недоступен до тех пор, пока измененные данные не будут вновь оттранслированы.

3. «Ввод названий узлов».

Данная функция доступна только при редактировании параметров узлов и дает возможность пользователю редактировать названия узлов.

Ее вызов осуществляется щелчком левой кнопкой мыши на соответствующей радиокнопке панели инструментов. В нижней части экрана непосредственно над панелью состояния выводится немодальное окно диалога, вид которого представлен на рис. 3.13.



Рис. 3.13. Вид немодального диалога «Ввод названий узлов».

Размеры оконного редактора соответствующим образом корректируются. В случае, если при отображении немодального диалога «Ввод названий узлов» оконному редактору останется недостаточно места для отображения хотя бы одной записи, выводится соответствующее сообщение, немодальный диалог не создается, а вызов функции игнорируется.

Оконная процедура диалога содержится в модуле ***NameUzlMod***.

В окно диалога выводится номер узла, соответствующего записи, на которой установлено окно строкового редактора, и название этого узла (не более 50 символов). При перемещении строкового редактора по записям эта информация обновляется.

Для редактирования названия узла необходимо щелкнуть левой кнопкой мыши на окне редактирования немодального диалога и при появлении в нем курсора отредактировать текст окна. Чтобы вернуть курсор строковому редактору, достаточно щелкнуть левой кнопкой мыши над рабочей областью оконного редактора.

Все время, пока активен режим ввода названий узлов, соответствующая радиокнопка на панели инструментов остается отмеченной. При повторном ее нажатии осуществляется выход из этого режима: считывается и запоминается содержимое окна редактирования, окно немодального диалога уничтожается, корректируются размеры оконного редактора, а радиокнопка разотмечается.

4. «Отметить строку (строки)».

Для удобства работы пользователя реализована возможность отметки строки (строк). Маркировать можно отдельную строку или несколько строк. В последнем случае при использовании «горячей клавиши» (ALT+L) или всплывающего меню оконного редактора отмечаются верхняя и нижняя строки в произвольной последовательности, допускается расширение отмеченного блока вверх и вниз.

Реализована возможность маркировки строки (строк) с помощью мыши путем ее перемещения с нажатой левой кнопкой. В момент отпускания кнопки операция выделения прекращается.

Отмеченная строка (строки) выводится на экран инвертированной.

5. «Отметить блок».

Для удобства работы пользователя реализована возможность отметки блока с помощью клавиатуры (используется «горячая клавиша» ALT+B) и с помощью мыши (используется двойной щелчок левой кнопкой или всплывающее меню оконного редактора). Маркировать можно отдельное поле или группу (блок) полей; в последнем случае отмечается левый верхний и правый нижний угол в произвольной последовательности. Допускается расширение отмеченного блока вверх и влево, вниз и вправо.

При отметке начала блока (к.-л. поля) оно инвертируется, а на панель состояния выводится надпись «Отметьте конец блока». При отметке конца блока отмеченная группа полей выводится на экран инвертированной, а надпись в панели состояния стирается.

6. «Снять маркировку».

Снять маркировку уже отмеченного блока или строки (строк) можно с помощью клавиатуры («горячая клавиша» ALT+U) и с помощью мыши (используется щелчок левой кнопкой мыши над рабочей областью оконного редактора или всплывающее меню оконного редактора).

7. «Поиск по шаблону вниз / вверх».

Данная функция облегчает и ускоряет поиск нужного параметра.

Шаблон поиска вводится в окно редактирования на панели инструментов (см. п. 3.4). Затем с помощью соответствующих органов панели инструментов («поиск по шаблону вниз», «поиск по шаблону вверх»), которые продублированы «горячими клавишами» (ALT+P и CTRL+P соответственно) и всплывающим меню оконного редактора, производится поиск.

Поиск ведется по столбцам, начиная с поля, на котором находится окно строкового редактора. «Листание» вверх / вниз производится автоматически, если это необходимо. Если поиск успешный, то окно строкового редактора устанавливается на поле, найденное первым; в противном случае пользователь информируется о том, что поиск неуспешен, а местоположение окна не изменяется.

Введенный последним шаблон сохраняется неизменным до тех пор, пока не будет введен новый.

**4. Разработка мероприятий по безопасной эксплуатации ПЭВМ**

Государственный комитет санитарно-эпидемиологического надзора Российской Федерации осуществляет разработку и надзор за исполнением федеральных санитарных правил, норм и гигиенических нормативов, которые:

◊ содержат нормативные акты, устанавливающие критерии безопасности и (или) безвредности для человека факторов среды его обитания и требования к обеспечению благоприятных условий его жизнедеятельности;

◊ обязательны для соблюдения всеми государственными органами и общественными объединениями, предприятиями и иными хозяйствующими субъектами, организациями и учреждениями, независимо от их подчиненности и форм собственности, должностными лицами и гражданами.

Федеральные санитарные правила и нормы в числе прочих включают в себя раздел «Гигиенические требования к видеодисплейным терминалам (ВДТ) и ПЭВМ и организация работы с ними», на основе которого разрабатываются мероприятия по безопасной эксплуатации ПЭВМ.

Рассмотрим некоторые наиболее важные с точки зрения безопасной эксплуатации ПЭВМ требования, устанавливаемые федеральными санитарными правилами и нормами.

**4.1 Требования к ВДТ и ПЭВМ**

Конструкция ВДТ и ПЭВМ должна обеспечивать соответствие санитарно-гигиеническим нормам следующих параметров:

◊ мощность экспозиционной дозы рентгеновского излучения;

◊ уровень ультрафиолетового излучения на рабочем месте пользователя;

◊ напряженность электромагнитного поля на рабочем месте пользователя по электрической и по магнитной составляющей;

◊ напряженность электростатических полей на поверхностях ВДТ;

◊ уровень звука при работе ВДТ и ПЭВМ и при включении их в сеть.

Конструкция ВДТ ПЭВМ должна предусматривать:

◊ возможность поворота корпуса в горизонтальной и в вертикальной плоскостях с фиксацией в заданном положении;

◊ антибликовое покрытие экрана и другие конструктивные решения для обеспечения оптимального качества изображения при наличии внешних источников освещения.

Конструкция клавиатуры должна предусматривать:

◊ исполнение в виде отдельного устройства;

◊ опорное приспособление, позволяющее изменять угол наклона поверхности клавиатуры.

**4.2 Требования к помещениям для эксплуатации ВДТ и ПЭВМ**

Помещения с ВДТ и ПЭВМ должны иметь естественное и искусственное освещение. Естественное освещение должно осуществляться через светопроемы, ориентированные преимущественно на север и северо-восток и обеспечивать нормированные для различных климатических поясов значения коэффициенты естественной освещенности (КЕО).

Расположение рабочих мест с ВДТ и ПЭВМ в цокольных и подвальных помещениях допускается только по согласованию с органами Государственного санитарно-эпидемиологического надзора.

Звукоизоляция ограждающих конструкций помещений с ВДТ и ПЭВМ должна отвечать гигиеническим требованиям и обеспечивать нормируемые параметры шума.

Помещения с ВДТ и ПЭВМ должны оборудоваться системами отопления, кондиционирования воздуха или эффективной приточно-вытяжной вентиляцией.

Поверхность пола в помещениях эксплуатации ВДТ и ПЭВМ должна быть ровной, без выбоин, нескользкой, удобной для очистки и влажной уборки, обладать антистатическими свойствами.

**4.3 Требования к защите от шума и вибрации**

В производственных помещениях, в которых работа на ВДТ и ПЭВМ является основной (диспетчерские, операторские, расчетные, кабины и посты управления, залы вычислительной техники и др.), в дошкольных и всех учебных помещениях с ВДТ и ПЭВМ фоновый уровень шума не должен превышать 40 дБА, (при работе систем воздушного отопления, вентиляции и кондиционирования – 35 дБА), а во время работы на ВДТ и ПЭВМ 50 дБА. Вибрация также не должна превышать нормируемых значений.

**4.4 Требования к освещению помещений и рабочих мест с ВДТ и ПЭВМ**

Искусственное освещение в помещениях эксплуатации ВДТ и ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, допускается применение системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Искусственное освещение должно обеспечивать на рабочих местах с ВДТ и ПЭВМ в производственных, административно-общественных, дошкольных и учебных помещениях освещенность не ниже нормируемых значений.

В качестве источников света при искусственном освещении должны применяться преимущественно люминесцентные лампы типа ЛБ. При устройстве отраженного освещения в производственных и административно-общественных помещениях допускается применение металлогалогенных ламп мощностью до 250 Вт. Допускается применение ламп накаливания в светильниках местного освещения.

Общее освещение следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, преимущественно слева, параллельно линии зрения пользователей.

**4.5 Общие требования к организации и оборудованию рабочих мест с ВДТ и ПЭВМ**

Рабочие места с ВДТ и ПЭВМ должны размещаться в помещениях с естественным освещением при ориентации оконных проемов на север или северо-восток.

Рабочие места с ВДТ и ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, следует изолировать друг от друга перегородками высотой 1,5–2,0 м.

Шумящее оборудование (АЦПУ и др.), следует размещать вне помещений с использованием ВДТ и ПЭВМ.

При конструировании оборудования и организации рабочего места пользователя ВДТ и ПЭВМ следует обеспечить соответствие конструкции элементов рабочего стола и их взаимного расположения, конструкции стула (кресла) гигиеническим и эргономическим требованиям с учетом характера выполняемой деятельности, комплексности технических средств, форм организации труда и основного рабочего положения пользователя.

Экран видеомониторов должен находиться от глаз пользователя на оптимальном расстоянии 700 мм, но не ближе 500 мм.

В помещениях с ВДТ и ПЭВМ ежедневно должна проводиться влажная уборка.

**4.6 Требования к организации режима труда и отдыха при работе с ПЭВМ и ВДТ**

Режимы труда и отдыха при профессиональной работе с ПЭВМ и ВДТ должны организовываться в зависимости от вида и категории трудовой деятельности. Виды трудовой деятельности разделяются на 3 группы:

◊ группа А – работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом;

◊ группа Б – работа по вводу информации;

◊ группа В-творческая работа в режиме диалога с ЭВМ.

Для видов трудовой деятельности устанавливается 3 категории тяжести и напряженности работы с ВДТ и ПЭВМ, которые определяются:

◊ для групп А и Б – по суммарному числу считываемых или вводимых знаков за рабочую смену;

◊ для группы В-по суммарному времени непосредственной работы с ВДТ и ПЭВМ за рабочую смену.

Продолжительность работы педагогов при ведении занятий с ВДТ и ПЭВМ во всех учебных заведениях не должна превышать 4 часов в день.

Продолжительность работы с ВДТ и ПЭВМ инженеров, обслуживающих занятия в кабинетах вычислительной техники или дисплейных классах высших учебных заведений, не должна превышать 6 часов в день.

Для обеспечения оптимальной работоспособности и сохранения здоровья профессиональных пользователей, на протяжении рабочей смены должны устанавливаться регламентированные перерывы. Время регламентированных перерывов в течение рабочей смены следует устанавливать в зависимости от ее продолжительности, вида и категории трудовой деятельности с ВДТ и ПЭВМ. Продолжительность непрерывной работы с ВДТ и ПЭВМ без регламентированного перерыва не должна превышать 2 часов.

Для профилактики зрительного утомления после каждых 25 минут работы следует выполнять комплекс упражнений для глаз.

Во время регламентированных перерывов, с целью снижения нервно-эмоционального напряжения, устранения влияния гиподинамии и гипокинезии, предотвращения развития познотонического утомления следует выполнять комплексы упражнений.

**4.7 Требования к организации медицинского обслуживания пользователей ВДТ и ПЭВМ**

Профессиональные пользователи ВДТ и ПЭВМ должны проходить обязательные предварительные (при поступлении на работу) и периодические медицинские осмотры в порядке и в сроки, установленные Министерством здравоохранения Российской Федерации и Государственным комитетом санитарно-эпидемиологического надзора Российской Федерации. К непосредственной работе с ВДТ и ПЭВМ допускаются лица, не имеющие медицинских противопоказаний.

Женщины со времени установления беременности и в период кормления ребенка грудью к выполнению всех видов работ, связанных с использованием ВДТ и ПЭВМ, не допускаются.

Медицинское освидетельствование студентов высших учебных заведений и учащихся средних специальных учебных заведений должно проводиться в соответствии с действующими нормативно-методическими документами Министерства здравоохранения Российской Федерации.

Медицинское освидетельствование детей дошкольного и школьного возраста проводится в порядке и в сроки, установленные Министерством здравоохранения Российской Федерации и Министерством образования Российской Федерации. [8]

**Заключение**

В ходе работы над проектом была дана математическая формулировка задачи оценивания и рассмотрены ее вычислительные аспекты, приведены концепции построения математического обеспечения информационно-вычислительной подсистемы, в соответствии с которыми должна формироваться единая математическая модель режима ЭЭС. Описаны также назначение и основные функции КП Компоновщик расчетных схем.

Рассмотрены требования, предъявляемые к системам подготовки, отладки и поддержания данных.

Приведен алгоритм формирования расчетной схемы, удовлетворяющий этим требованиям.

Описан пользовательский интерфейс КП Компоновщик расчетных схем в операционной системе MS DOS.

Реализованы основные элементы пользовательского интерфейса в операционной системе Windows: панель инструментов, панель состояния, строковый и оконный редакторы, модальные и немодальные диалоговые окна.

Рассмотрены некоторые наиболее важные с точки зрения организации безопасной эксплуатации ПЭВМ требования, устанавливаемые федеральными санитарными правилами и нормами.

Разработанные программы насчитывают порядка трех с половиной тысяч операторов.

**Список литературы**

1. Унароков А.А. Управление режимом энергосистемы в реальном времени. // Докторская диссертация. – М.: МЭИ, 1997 г.
2. Унароков А.А. Вычислительные аспекты оценки состояния ЭЭС. // Известия РАН. Энергетика, 1995 г., №2.
3. Унароков А.А. Математическое обеспечение информационно-вычислительной подсистемы: состав, концепции построения и технические характеристики. // Сборник научных трудов №9. – Смоленск: СФ МЭИ, 1996 г.
4. Петзолд Ч. Программирование для Windows 95; в двух томах. / Пер. с англ. – СПб.: BHV – Санкт-Петербург, 1997 г.
5. Бартеньев О.В. Современный Фортран. – М.:Диалог-МИФИ, 1998 г.
6. Гладков С.А., Фролов Г.В. Программирование в Microsoft Windows, в двух частях. Часть 1. – М.: Диалог-МИФИ, 1992 г.
7. Руководство по программированию в среде Microsoft Windows. – М.: Наука, 1991 г.
8. Федеральные санитарные правила, нормы и гигиенические нормативы. Гигиенические требования к видеодисплейным терминалам и персональным электронно-вычислительным машинам и организация работы с ними. – ГК СЭН РФ, М.: СанПин, 1994 г.

1. Более подробно немодальные, а также модальные окна диалога будут рассмотрены в п. 3.3. [↑](#footnote-ref-1)