Ш.37; ЄЯ/. 3.0£.C'Of-5>

5V.43. So.W.tf

ХЙРЬКОВСКЙЯ ГОС9ДАРСТВЕННАЯ АКАДЕМИЯ ПЛЕЗН0Д0Р01Н0Г0 ТРАНСПОРТА

на правах рукописи

ЗОРИНА ЕЛЕНА ИВАНОВНА

ТЕХНОЛОГИЧЕСКИЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМ УПРАВЛЕНИЯ ГОМОГЕННЫМИ ОБЪЕКТАМИ

(на примере микропроцессорной централизации)

05.13.09 — математическое *и* программное обеспечение, вычислительных мамин и систем

Диссертация

на соискание ученой степени кандидата технических наук

НАУЧНЫЙ РУКОВОДИТЕЛЬ
кандидат технических наук, доцент
Добрянский В.М.

Харьков 1996

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Применение микропроцессоров (МП) и микро-ЭВМ в локальных вычислительных системах обеспечивает достижение высоких технике-экономических показателей и расширение функциональных возможностей этих систем. Повышение степени интеграции МП, снижение их стоимости, повышение надежности, рост функциональных возможностей позволяют эффективно использовать их в системах железнодорожной автоматики.

В последние годы в ряде стран ведутся работы по созданию микропроцессорных систем управления стрелками и сигналами на железнодорожном транспорте (микропроцессорной централизации - МПЦ). ХарГАЖТ (ХИИТ) в рамках международного сотрудничества участвует в разработке структуры технических средств МПЦ и методов нормирования их надежности и безопасности. Полученные рекомендации являются обобщением опыта ХИИТа по разработке микропроцессорной централизации, завершившейся созданием действующего макета. Технологические методы проектирования программного обеспечения (ПО) МПЦ, использовавшиеся при создании макета, не удовлетворяют современным требованиям. Такие аспекты, как повышение надежности разрабатываемого ПО и снижение его стоимости, не рассматривались вообще. Разработки велись для конкретной железнодорожной станции и носили ''поисковый", экспериментальный характер без учета использования системы в будущем на иной станции. МПЦ же обладает рядом особенностей, которые позволяют однажды разработанную автоматизированную систему использовать для других станций. Поэтому актуальной является задача разработки единой для

всего множества станций технологии проектирования ПО, позволяющей автоматизировать процесс генерации ПО МПЦ.

Целью диссертационной работы является разработка технологических аспектов проектирования ПО систем управления (и МПЦ, в частности), ориентированных на класс объектов (железнодорожные станции) и обеспечивающих достижение требуемого уровня надежности ПО при снижении затрат на его разработку.

В соответствии с поставленной целью в работе решались следующие задачи:

анализ характеристик качества ПО;

оценка взаимозависимости надежности и стоимости ПО;

анализ и поиск путей снижения стоимости и повышения надежности ПО;

исследование существующих технологий проектирования ПО и определение характеристик, влияющих на эффективность технологии проектирования и количество этапов проектирования;

классификация объектов управления (ОУ) и введение понятия "гомогенных объектов управления";

анализ степени однородности (гомогенности) железнодорожных станций как ОУ;

разработка методики генерации ПО для систем управления (СУ) гомогенными объектами;

разработка схемы генерации ПО требуемой конфигурации СУ гомогенными объектами;

разработка технологии проектирования ПО микропроцессорной СУ стрелками и сигналами.

Объектом исследования являются железнодорожные станции и другие аналогичные по структуре объекты. Предметом исследо-

вания является система управления гомогенными объектами (МПЦ).

Теоретической основой исследования послужили научные труды и решения, принятые в области теории САПР, технологий проектирования ПО, теории надежности, качества программного продукта.

При введении понятия гомогенных объектов управления, характеристике принципов функциональной избыточности и функциональной избирательности, разработке схемы генерации ПО СУ гомогенными объектами использовались элементы теории множеств, алгебра логики, эвристические методы.

Научная новизна. Проблема создания ПО СУ сформулирована как задача поиска.таких технологий его проектирования, которые обеспечивали бы необходимый уровень надежности разработки при снижении стоимости технологических шагов. Выделены характеристики эффективности технологий проектирования ПО. Произведена классификация ОУ и введено понятие гомогенных ОУ. Предложена методика проектирования ПО для гомогенных ОУ. Разработана технология проектирования ПО МПЦ.

Практическая ценность работы состоит в том, что предложенная методика проектирования ПО СУ гомогенными объектами позволяет:

автоматизировать процесс генерации ПО МПЦ на основе разработки единой для всего множества станций технологии проектирования ПО;

реализовать такие технологические процессы, в которых большая часть технологических шагов-общая для всех станций;

обеспечить необходимый уровень надежности разрабатывав-

мого ПО при снижении его стоимости.

Разработан пакет программ проектировщикаописания станций.

Диссертация выполнена в соответствии с планом научно-исследовательских работ, проводимых в ХарГАЖТ (ХИИТ) по указанию МПС, а позднее - Министерства транспорта Украины в рамках международного сотрудничества. Отдельные проектные решения внелрены в НПО САУ (г.Харьков), 'Московском государс- - твенном университете путей сообщения, Международной академии электротехнических наук (г.Москва), институте Гипротранссигналсвязь (г. Санкт-Петербург).

По теме диссертации опубликовано 13 печатных работ. Результаты исследований отражены в отчетах о НИР.

Диссертационная работа состоит из введения, четырех глав, заключения, списка литературы и трех приложений.

Во введении обоснована актуальность рассматриваемых в диссертации проблем, приведена краткая аннотация работы, сформулированы цель и задачи исследования, раскрыта научная новизна и практическая значимость полученных результатов.

В первой главе рассматриваются технологические задачи создания программного обеспечения систем управления, выделяя методологические аспекты надежности и стоимости, пути снижения стоимости и повышения надежности программного обеспечения. Производится анализ технологий проектирования программного обеспечения с выявлением показателей, влияющих на эффективность технологии проектирования, конкретное количество ее этапов, формулируется основная цель исследования и пути ее решения.

Концептуальной основой построения эффективной технологии проектирования для функционально однородных объектов управление точки зрения соотношения между стоимостью и надежностью является вводимая во 2-ой главе математическая модель всего класса подобных объектов. Функционально однородные объекты названы гомогенными. На основании модели предложены' методика проектирования программного обеспечения систем управления гомогенными объектами,

. принципы технологии проектирования программного обеспечения систем управления гомогенными объектами, в которой большая часть технологических шагов - общая для выделенных объектов.

Третья глава посвящена разработке методов генерации программного обеспечения систем управления гомогенными объектами. Среди них выделяются:

принципы функциональной избыточности и функциональной избирательности,

лингвистические задачи генерации;

контроль достоверности программного обеспечения.

Четвертая глава посвящена практическому применению предложенных методов генерации программного обеспечения систем управления гомогенными объектами при проектированиии микропроцессорной системы управления стрелками, и сигналами на железнодорожном транспорте. Разработана система проектирования подсистемы отображения ^информации.

В заключении приведены основные выводы.

В приложениях приведены материалы, иллюстрирующие работу фрагментов прикладного программного обеспечения.

1. ТЕХНОЛОГИЧЕСКИЕ ПРОБЛЕМЫ СОЗДАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1.1. Надежность и стоимость программного обеспечения

Развитие программирования за последние годы привело к разработке методов и средств, позволяющих создавать программы для разных областей применения и типов вычислительных машин. Рост доверия к программам-и к их возможностям выполнять различные логические и вычислительные функции не только увеличил объем разработки, но и значительно повысил важность и ответственность выполняемых ими. функций. Рост значения результатов функционирования пакетов прикладных программ и сложных комплексов программ управления и обработки информации повысило интерес пользователей и разработчиков к анализу качества создававемых и эксплуатируемых программ.Отторжение программ от их первичных создателей привела к формированию понятия программного продукта. Этот продукт является результатом нового вида современного промышленного производства. Из объекта научного творчества и произведения искусства отдельных программистов программы превратились в объект планомерной разработки, эксплуатации и сопровождения большими коллективами специалистов.

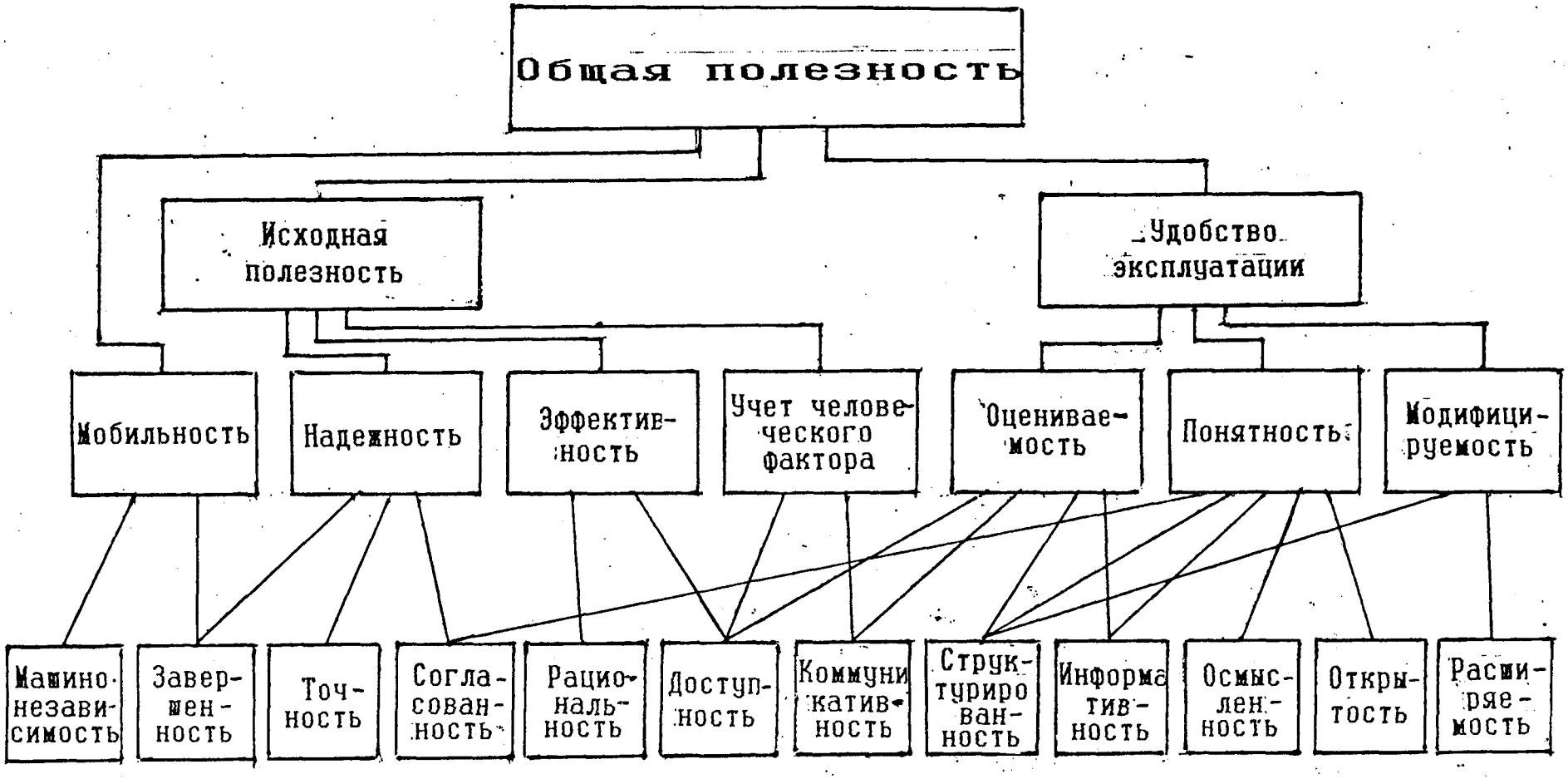
Для превращения программирования в современную инженерную дисциплину проектирование программного обеспечения (ПО) следует осуществлять программными и научными методами, с обязательным применением различных способов измерения харак*

теристик программ и установлением между ними причинно-следственных связей. Для этого необходимо научиться измерять и прогнозировать характеристики качества программ, а также изучать зависимость этих характеристик от различных параметров. Комплексы программ характеризуются конкретными показателями качества (рис.1).

При оценке качества ПО определяющим фактором является многообразие интересов пользователя. Именно по этой причине невозможно предложить какую-либо единственную универсальную меру качества программных средств. Здесь необходимо множество характеристик, охватывающих целый спектр желаемых свойств. Каждая такая характеристика должна подразделяться на отдельные показатели, оценивающую ту или иную ее сторону. Возникает необходимость построения иерархической системы все более детализированных характеристик, в которой каждый вышестоящий уровень приближается к реальным нуждам пользователей, а каждый нижестоящий уровень постепенно ведет к получению числовых оценок соответствующих свойств.

Выделение множества элементарных характеристик, приведенных на рис.1, создает основу для определения количественных показателей, которые могли бы применяться для оценки степени, в которой ПО обладает как указанными элементарными свойствами, так и характеристиками более высоких уровней. И те, и другие должны участвовать в формировании оценки общей полезности ПО (оценки высшего уровня).

Иерархическое дерево свойств программного продукта



м

Характеристики, расположенные на нижнем уровне дерева, представляют собой множество элементарных свойств, которые имеют фундаментальные отличия друг от друга и группируются в наборы соответственно условиям, необходимым для существования конкретных характеристик промежуточного уровня. Так, программа, которая не зависит от конкретной машины, но не обладает свойствами точности, согласованности, рациональности, доступности, коммуникативности, информативности, структурированности, осмысленности, открытости и расширяемости, все же будет удовлетворять критериям мобильности.

На нижнем уровне элементарные характеристики - следующие [73:

Программный продукт (ПП) обладает свойством машинонезависимости, если входящие в него программы могут выполняться на вычислительной машине иной конфигурации, чем та, для которой они предназначены.

Завершенным считается программный продукт, если в нем присутствуют все необходимые компоненты, позволяющие пользователю самостоятельно эксплуатировать программу.

Программный продукт обладает свойством точности, если выдаваемые им результаты имеют точность, достаточную с точки зрения основного их назначения.

. Программный продукт характеризуется свойством внутренней согласованности, если он всюду содержит единую нотацию, терминологию и символику. ПII обладает свойством внешней согласованности, если можно проследить его соответствие требованиям заказчика.

Доступным будем считать ПП, если допускается селективное

использование отдельных его компонент.

Коммуникативностью характеризуется программное изделие, которое дает возможность легко описывать входные данные и выдает информацию, форма и содержание которой просты для понимания и несут полезные сведения.

Ш обладает свойством информативности, если содержит информацию, необходимую ***и*** достаточную для понимания читающим лицом назначения программных средств, принятых допущений, существующих ограничений, исходных данных, результатов.

Осмысленностью характеризуется программный продукт, если его документация не содержит избыточной информации.

Открытым является ПО, если его функции и назначение соответствующих операторов легко понимаются в результате чтения текста программы.

Программное изделие расширяемо, если позволяет увеличивать при необходимости, объем памяти для хранения данных, или расширения вычислительных функций отдельных модулей.

На промежуточном уровне выделяются следующие характеристики.

ПП обладает свойством мобильности, если он может легко и эффективно использоваться для работы на ЭВМ иного типа, чем та, для которой он предназначен.

.Программное изделие надежно, если можно ожидать, что оно будет удовлетворительно выполнять необходимые функции [7]. Программный продукт является эффективным, если он выполняет требуемые функции без излишних затрат ресурсов.

. Программа учитывает человеческий фактор, если она способна выполнять свои функции, не требуя излишних затрат време-

ни со стороны пользователя, неоправданных усилий пользователя по поддержанию процесса функционирования программ и без ущерба для морального состояния пользователя.

Ш обладает свойством оцениваемости, если это свойство позволяет установить критерий приемлемости ПП для конкретного применения и обеспечивает возможность оценки качества функционирования программных средств.

Программное изделие понятно в той степени, в которой оно позволяет оценивающему лицу понять назначение программных средств.

ПП характеризуется модифицируемостью, если он имеет структуру, позволяющую легко вносить требуемые изменения.

Исходная полезность характеризует только пригодность программы для работы в исходном ее виде и на требуемой ЭВМ. Программный продукт удобен в эксплуатации, если предусматривается возможность его обновления в соответствии с новыми требованиями.

Общая полезность показывает в какой мере данная программа позволяет вносить изменения и использовать ее в условиях, отличных от известных.

'■ Анализ характеристик качества программных средств показывает, что многие из них противоречивы:

увеличение эффективности, как правило, становится возможным за счет ухудшения мобильности, точности, понятности и удобства эксплуатации;

повышение точности отрицательно сказывается на мобильности;

достижение высокой степени осмысленности может вступать

в конфликт с ограничениями на открытость.

Для устранения таких противоречий необходимо ввести взаимосвязанные характеристики качества ПО и разработать систему показателей, позволяющих обнаружить аномалии соответствующих характеристик. Введение совокупности элементарных характеристик, представленных на рисунке 1, не приводит к нарушению правильности данных определений свойств ПП, а только создает иерархическую структуру свойств ПП. Так, информативность становится одним из аспектов оцениваемости, понятности, модифицируемости, которые в свою очередь характеризуют различные стороны удобства эксплуатации.

Таким образом,к элементарным характеристикам отнесены 12 свойств, а 7 свойств представляют собой композицию выделенных элементарных характеристик.

Опыт разработки крупных программных комплексов выдвинул проблему надежности их функционирования, как одного из показателей качества, в разряд самых актуальных [26]. В частности, проблема обеспечения требуемого уровня надежности в управляющих вычислительных системах является наиболее сложной. Сложность проблемы обусловлена появлением ряда специфических особенностей, отличающих ее от проблемы надежности традиционных систем управления. В традиционных системах проблема надежности замыкается только в пределах надежности аппаратуры, в то время как в управляющих вычислительных системах, кроме надежности аппаратуры, огромную роль играет надежность ПО.

ОРИПРТТИР ттПЧЯТНЯ тгя ттО чгалгфтл тупжягтпотглпт» ттпгпо»,»» баотпп- V xlvJxlЛ X Jrxrfl XX<U,MkyrfAU.XV/4/XKX XVWXYXXXWJLWXtWiyXJ XX^/lYX £/UXYXXYI V/C4

ются на понятиях теории.надежности, первоначально развившей-

ся для аппаратурных комплексов. При этом фундаментальным является понятие надежности как ''свойство объекта сохранять во времени в. установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях применения.'' (ГОСТ 27.002-83).

Надежность ПО - это вероятность того, что программа какой-то период времени будет работать без сбоев с учетом степени их влияния на выходные результаты [29]. Надежность программы - это ее свойство сохранять во времени способность верно выполнять свои функции при наличии возможных причин отказов программ, описанных в техническом задании на программирование [86].

Согласно этим определениям, надежность ПО - свойство сохранять работоспособность на некотором отрезке времени или при выполнении некоторого объема работы. В соответствии с этими определениями надежность является внутренним свойством системы, заложенным при ее изготовлении и проявляющимся при эксплуатации. Это свойство проявляется только во времени, поэтому без длительного наблюдения и учета значений времени нельзя сделать заключения о надежности системы.

Надежность ПО существенно, отличается от надежности аппаратуры. В отличие от аппаратуры, программы не изнашиваются, поломка программы невозможна. Ненадежное ПО - следствие ошибок, внесенных в процессе его разработки. Надежность аппаратуры определяется во многом случайными сбоями, а надежность ПО - скрытыми в нем ошибками. Сбои аппаратуры не зависят от обрабатываемых системой данных. Проявление же ошибок в программе зависит от входных данных. Кроме того, час-

тота сбоев существенно зависит от времени. Частота же, с которой обнаруживаются ошибки в ПО, является функцией входных данных и состояния системы.

Таким образом, приведенные выше определения не отражают истинной природы надежности ПО. Надежность ПО не является внутренним свойством программы, она связана с тем, как программа используется. Проявление ошибок зависит от исходных данных на входе в программу. Даже крупный просчет в проектировании может оказаться не слишком заметным для пользователя. С другой стороны, тривиальная ошибка может иметь катастрофические последствия.

Приведенные характеристики надежности ПО отражены в определении Майерса [88], который под надежностью ПО понимает вероятность проявления ошибки в нем. с учетом ее стоимости для пользователя. Этого определения мы и будем придерживаться в дальнейшем.

Стоимость ошибки определяется степенью ее влияния на процесс управления. Так, пусть в программном обеспечении имеются две ошибки: в подпрограмме формирования маршрута (для микропроцессорной системы управления стрелками и сигналами) и в программе вывода сообщений оператору на экран диалогового дисплея. Первая проявляется в том, что неверно устанавливается маршрут, а вторая - в грамматической ошибке в тексте сообщения. Обе ошибки формально равнозначны. Однако проявление первой из них может привести к катастрофическим последствиям, а вторую можно и не заметить (не очень грамотный оператор может посчитать, что так и надо). Поэтому вклад каждой из ошибок в снижение надежности ПО неравноценен.

Стремление к максимально ' возможному уровню надежности ведет к повышению стоимости разработки. При этом зависимость стоимости и надежности имеет нелинейный характер и, начиная с некоторого уровня, стремление к повышению надежности приводит к затратам, соизмеримым со стоимостью всего предыдущего этапа разработки . Для большинства систем обработки данных и информационно-поисковых систем в подобных случаях возможен отказ от дальнейших усилий по повышению надежности разрабатываемого программного изделия. В этом случае этап эксплуатации ПО можно рассматривать как продолжение тестирования.

Выявленные в процессе эксплуатации ошибки затем идентифицируются и устраняются (на технологическом уровне эти процедуры составляют этап сопровождения ПО) и, таким образом, с течением времени эксплуатации ПО его надежность повышается (рис. 2).

В отличие от систем обработки данных и информационно-поисковых систем подобный прием для большинства управляющих вычислительных систем неприемлем. На 1-ый план выдвигаются допросы надежности разрабатываемого ПО. Ошибки в ПО здесь могут вести к катастрофическим последствиям. Зто обстоятельство определяет максимальное использование методологических и технологических ресурсов в интересах повышения надежности независимо от затрат. Для ПО таких управляющих систем задается минимально допустимый уровень надежности, а тестирование и отладка ведутся до тех пор, пока этот уровень не будет достигнут. В этом случае стоимость ПО как изделия будет зависеть от технологии его проектирования.

Зависимость надежности ПО от времени эксплуатации

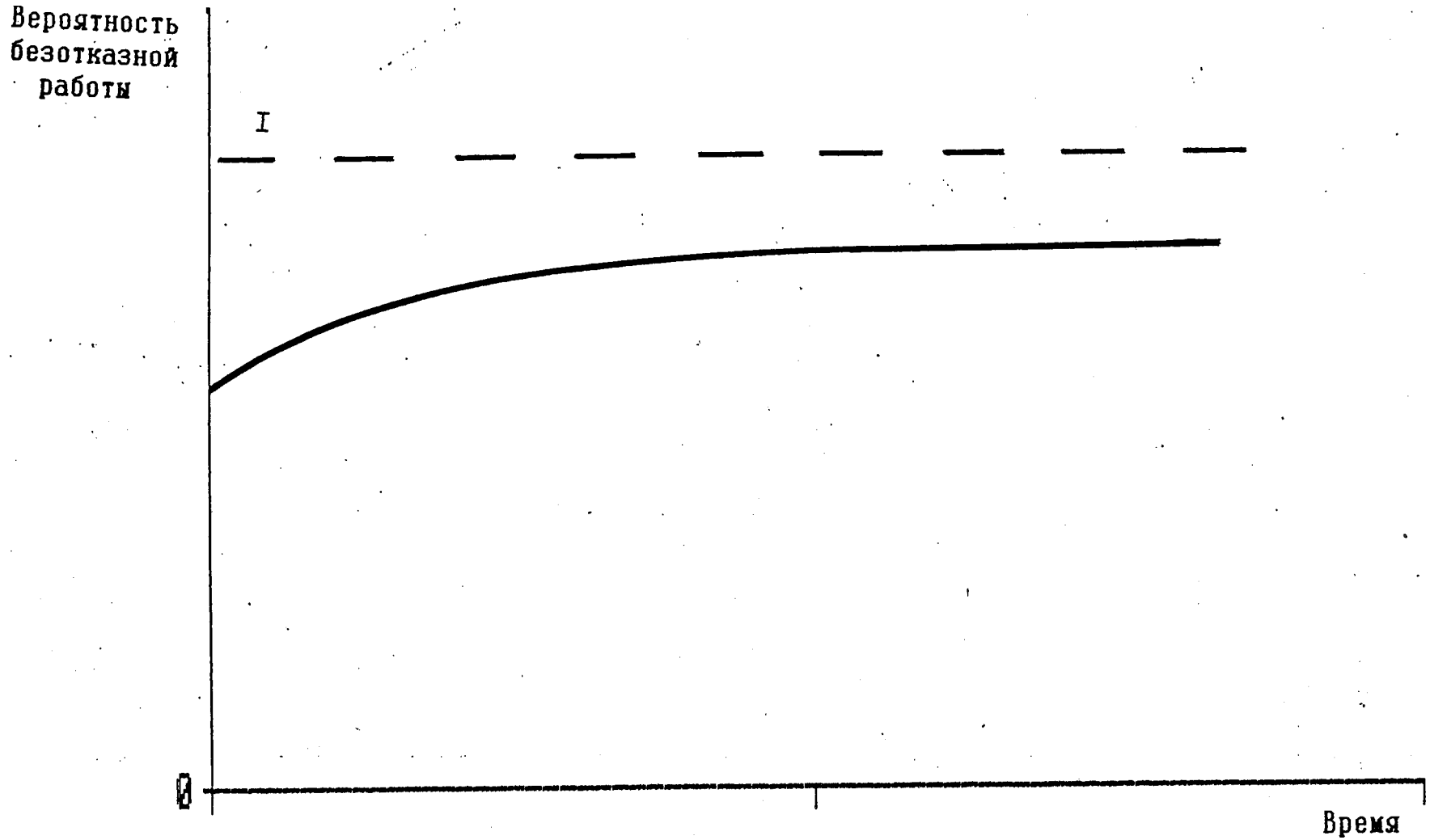


Рис. 2

Требуемый уровень надежности может достигаться различными технологиями, отличающимися друг от друга характером зависимости надежности ПО и затрат на его создание. На рис. 3 изображена зависимость затрат на создание ПО от вероятности безотказной работы, где {T_1,T_2,T_3,T_4> - возможные технологии программирования. Будем считать, что каждая технология состоит из конечного множества А технологических шагов и соответсвующего ему множества В средств их реализации. Другими словами, множество А представляет собой множество технологических шагов на макроуровне. Множество В -. не что иное, как множество шагов реализации каждого шага макроуровня на микроуровне. Тогда произвольная модификация любого из множеств (А или/и В), удовлетворяющая конечной цели проектирования, порождает новую технологию. Множество всех возможных технологий С есть произведение множеств технологических шагов и средств их реализации, то есть

$$C = A * B, \text{ где } A = \{ A_г, A_2, \ldots, A^{\wedge}\},$$

$$B = \{ B_t, в_2, \ldots, в_п>.$$

Задача выбора из множества С технологии, обеспечивающей минимум прямых и косвенных затрат на создание ПО является чрезвычайно важной. Однако на практике построение множеств А и В невозможно. Поэтому актуальным является поиск отдельных компонентов множеств А и В, отличных от уже известных и обеспечивающих уменьшение затрат и требуемый уровень надежности. Таким образом, проблема создания ПО для управляющих систем может быть сформулирована как задача поиска таких технологий его разработки, которые бы обеспечивали достижение требуемого уровня надежности при уменьшении затрат.

Зависимость затрат на создание ПО от вероятности безотказной работы

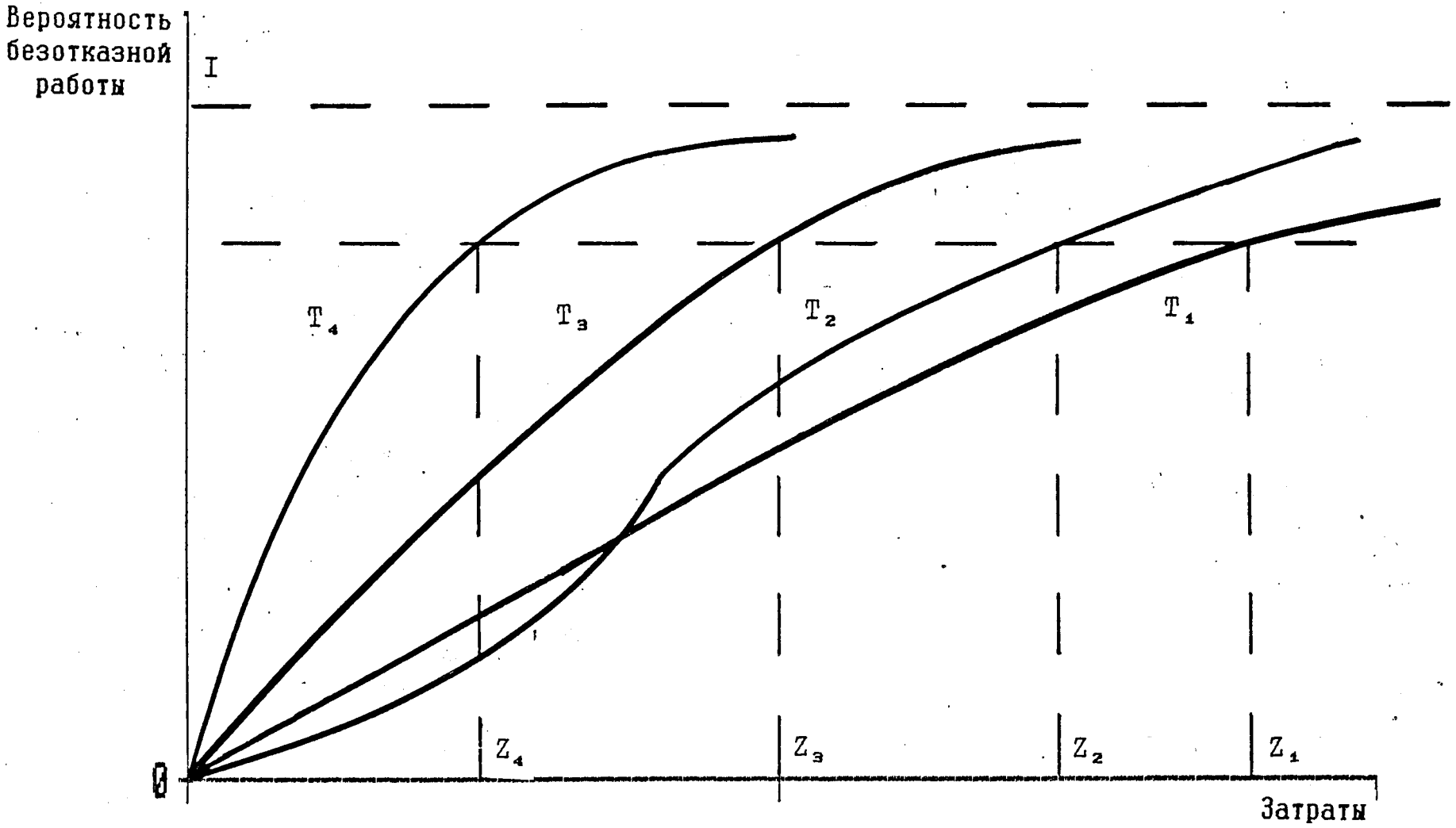


Рис. 3

1.2. Пути снижения стоимости и повышения надежности программного обеспечения

1.2.1. Анализ технологий проектирования программного обеспечения

Традиционная технология проектирования (ТП) складывалась в научных учреждениях с середины сороковых годов до конца шестидесятых тысячами программистов как технология талантливых ремесленников, в которой научная составляющая была незначительной. Такие проблемы как понятность, надежюсть, удобство эксплуатации программ по существу не затрагивались и, начиная с некоторого момента, возникла необходимость в поиске новых технологических принципов. В начале 70-х годов одновременно со структурным программированием появилась технология программирования как наука. С тех пор теория проектирования ПО представляет собой подход, охватывающий методологию программирования, проблемы обеспечения надежности программ, оценки рабочих характеристик и качества проектов, вопросы управления проектированием систем программного обеспечения, а также средства и стандарты разработки программ [17].

Все используемые в настоящее время виды технологии программирования (структурное программирование, HIPO-диаграммы и т.д.) в основном применяются на этапе собственно кодирования и иногда затрагивают этап.проектирования. Остальные же весьма важные этапы технологий программирования разрабатываются скорей интуитивно, без применения эффективных методов их реализации, что снижает надежность и .увеличивает стоимость ПО ' в целом.

Под ТП будем понимать науку об эффективных способах (приемах и процедурах) проведения процесса программирования, обеспечивающего в заданных условиях получение программной продукции с заданными свойствами [13]. Весь процесс создания ПО может быть разделен на ряд независимых технологических этапов (уровней, фаз) (рис.4): внешнее проектирование, проектирование, кодирование, тестирование и отладка, сопровождение. Уровни могут пересекаться и конкретное их число и содержание определяются целями создания системы управления.

Фаза внешнего проектирования (постановка задачи, внешние спецификации, системный анализ) определяет все внешние связи и выполняемые функции. Основная цель этапа заключается в выборе и анализе требований к ПО, описании целей разработки, состава и характеристик потоков входной-выходной информации и т.д. Внешнее проектирование проявляется в виде внешних спецификаций - законченного и строгого изложения о том, что представляет собой программное изделие, для чего оно предназначено. Решение задачи в терминах спецификаций является основным итогом фазы. Хорошо продуманные спецификации жизненно важны для обеспечения надежности и качества программ, В большинстве случаев ошибки в системе - результат неполных или противоречивых спецификаций. Из-за них в работе возникает до 2/3 ошибок [881. Причем, чем раньше обнаружена ошибка, тем меньше стоимость ее исправления. На рис.5 представлена зависимость относительных затрат на корректировку ошибок от этапов разработки программ [17]. Стоимость обнаружения и исправления каждой ошибки быстро (в 10-100 раз) возрастает по мере приближения к завершающим этапам проектирования комплекса программ.

Технологические уровни создания ПО

Внешнее проектирование

Про ектирование

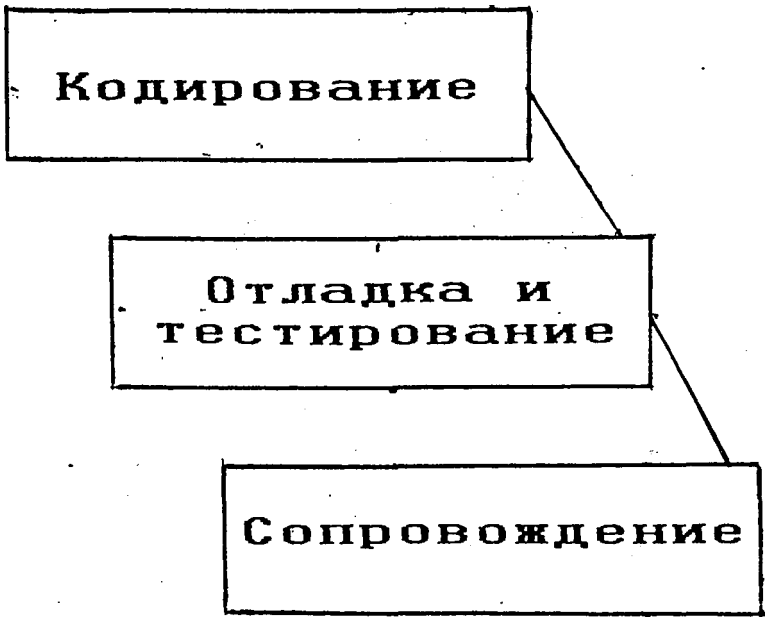


Рис /4

График зависимости стоимости ПИ $ и вероятности исправления ошибки Р от времени ее обнаружения

P S -I

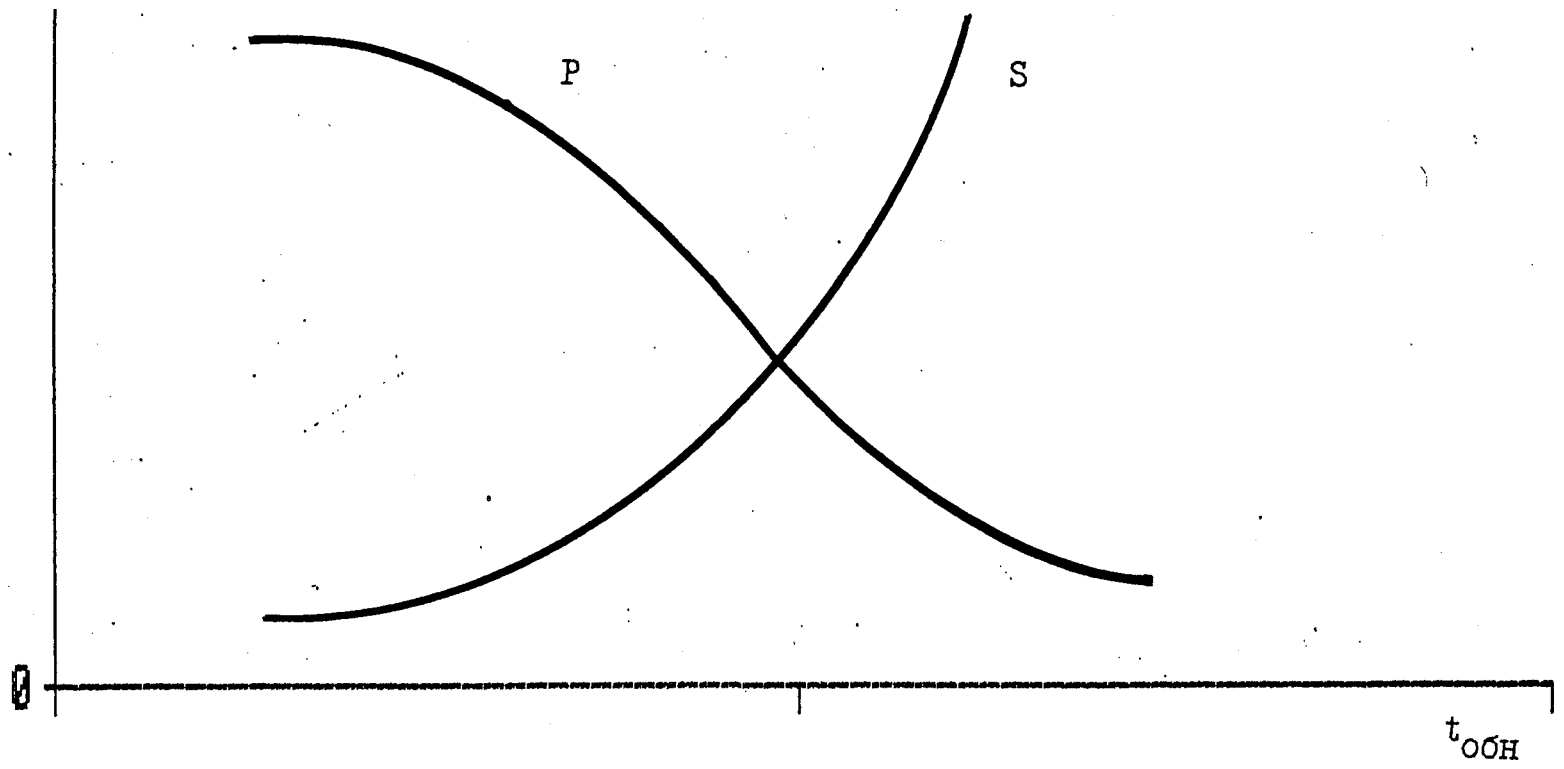


Рис. 5 •

м
сп

Поэтому можно говорить об ожидаемом повышении надежности ПО в целом, а следовательно, и снижении стоимости разработки благодаря применению новых эффективных методов фазы внешнего проектирования. Внешние спецификации должны соответствовать следующим требованиям: простота, краткость, точность, удобочитаемость, полнота. Для повышения надежности и уменьшения затрат на этом этапе применяются системное моделирование и имитация.

Проектирование можно определить как этап "внутреннего проектирования". Если во внешних спецификациях описывается, что делает ПО, то во внутренних - каким образом оно должно быть сконструировано и как достигаются установленные для него цели и требования. На этапе внутреннего проектирования осуществляется описание внутренней структуры ПО и стратегия его разработки. Основным итогом этой фазы являетсяполучение проекта в форме текста на естественном языке, блок-схем, решающих таблиц, текстов на алгоритмическом языке или в какой-нибудь другой форме. Методы повышения надежности на данном этапе состоят в проверке проекта, что экономит больше времени и средств, чем на нее затрачивается, благодаря раннему обнаружению и устранению ошибок, исправление которых в дальнейшем будет обходиться очень дорого.

Программирование (кодирование) - фаза реализации проекта. Решения, принятые на предыдущем этапе, преобразуются в форму, доступную для ЭВМ. Проблема надежности на этом этапе решается выбором наиболее приемлемого способа программирования. Наиболее распространенным в настоящее время являются

методы модульного программирования, структурного программирования, программирования сверху-вниз и ряд других.

Тестирование и отладка играют очень важное значение для повышения надежности ПО. В общем времени разработки программ отладка и тестирование занимают по оценкам специалистов от 50% до 60% [871. Авторы, занимающиеся вопросами разработки ТП, дают различные определения этапу тестирования и отладки. В работе под тестированием будет пониматься процессвыполнения программ с намерением найти ошибку. Основной целью тестирования является выявление максимального числа ошибок. Отладка же направлена на установление точной природы известной ошибки, а затем исправление ее. Результаты тестирования являются исходными данными для отладки (рис.6). Весь этап имеет итеративный характер, причем число итераций (соответственно - стоимость этапа) зависит от количества ошибок, не выявленных на предыдущих этапах, и требуемого уровня надежности ПО. Интенсивность появления ошибок уменьшается по мере их обнаружения и устранения. То есть, чем больше количество итераций, тем выше вероятность безошибочной работы ПО. На рис.7 приведена зависимость интенсивности программных ошибок от продолжительности отладки. Весьма важным при разработке ПО является определение момента окончания тестирования и отладки - момента, когда II0 становится пригодным для эксплуатации.- Методы установления этого момента можно разделить на два основных вида: простая оценка оставшейся работы - тестирование прекращается, когда проверено выполнение всех требований к качеству (все тестовые примеры успешно прошли); методы количественной оценки - используют модели надежности ПО.

Сже&а взаимосвязи

» I

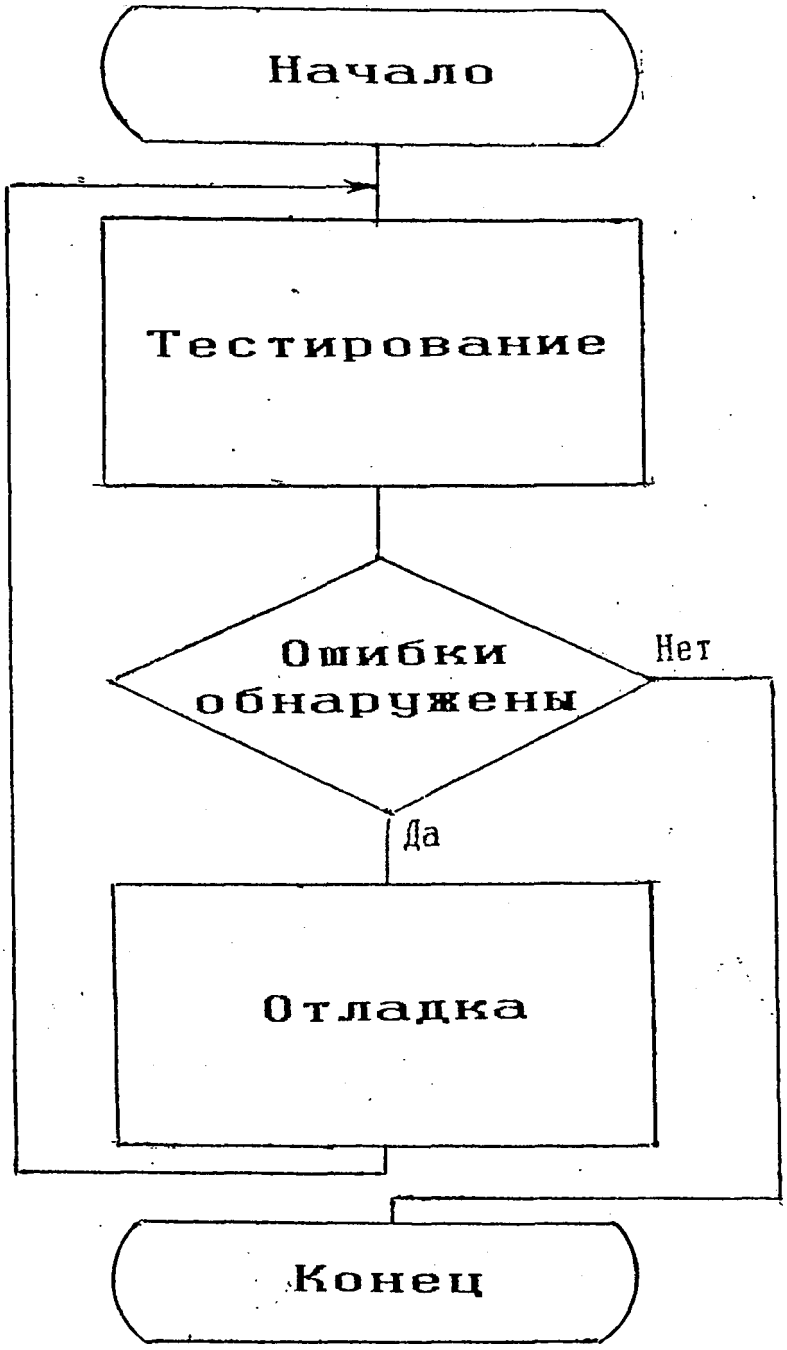


Рис _6

Зависимость интенсивности обнаружения программных ошибок

от продолжительности отладки

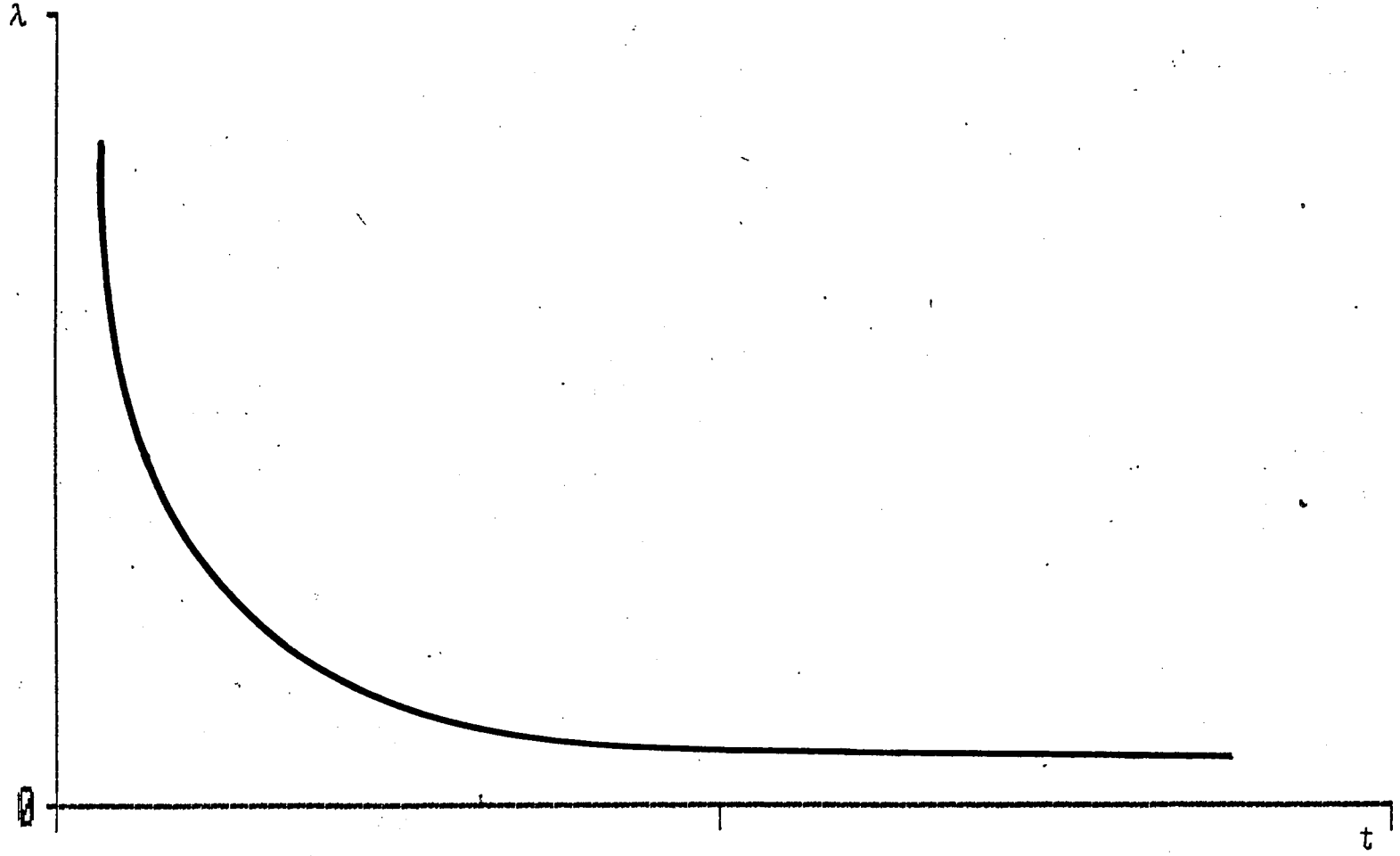
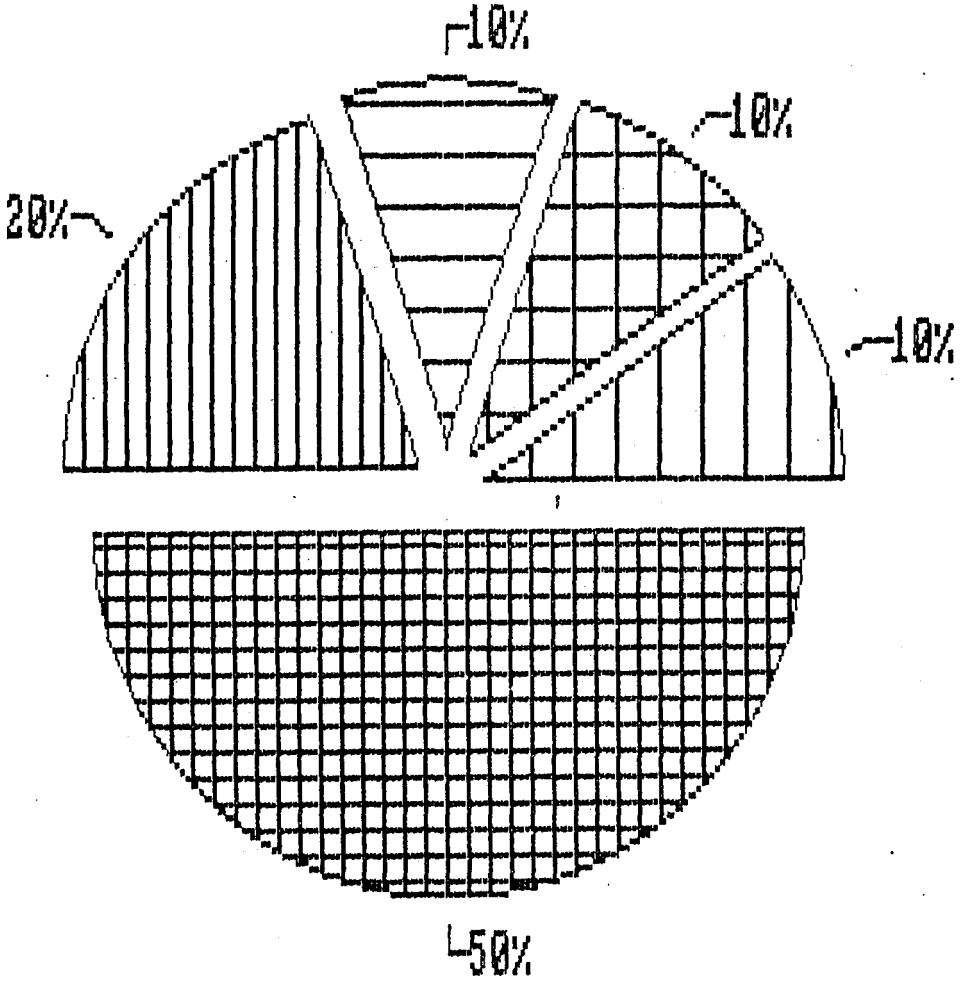


Рис.7

Накопленное за определенный период число ошибок сравнивается с числом ошибок, полученным на модели. Если реальные данные подтверждают результаты моделирования, то момент окончания тестирования может быть оценен с определенной достоверностью.

Сопровождение ПО представляет собой процесс по исправлению ошибок в нем и внесению необходимых изменений, выявленных на стадии эксплуатации. Методы повышения надежности на предыдущих этапах проектирования должны быть направлены на минимизацию затрат на этапе сопровождения, так как в стоимостном выражении этап составляет 50% от общей стоимости разработки (рис.8). Здесь необходимо акцентировать внимание на том, что тип создаваемой системы оказывает значительное влияние на ТП, на продолжительность этапов разработки. Действительно, в реальных условиях этап сопровождения рассматривается как продолжение этапа тестирования и отладки. Для большинства же управляющих систем на первый план выдвигаются вопросы надежности разрабатываемого ПО. Ошибки в ПО здесь недопустимы. Таким образом, для- блокирования негативных последствий в таких управляющих системах можно говорить о повышении роли этапов внешних спецификаций, проектирования, кодирования, отладки и тестирования, и о значительном сокращении этапа сопровождения, а в некоторых случаях даже его ликвидации, что должно существенно уменьшить стоимость разрабатываемого ПО в целом.

Распределение стоимости ЛО по Фазам разработки



Внешнее проектирование

Проектирование

Кодирование

Отладка

Сопровождение

Рис, 8

Кроме того, в схеме, представленной на рис. 4, отсутствует этап документирования, который во многих источниках выделяется как отдельный этап. С целью уменьшения стоимости ПО процесс документирования необходимо начинать с первого этапа разработки ПО (этапа внешнего проектирования) и выполнять на всех последующих этапах.На каждом из этапов будет подготавливаться необходимая документация. Таким образом, к завершению процесса разработки будет собран весь необходимый набор документации.

Технология проектирования ПО должна представлять набор технологических шагов и средств их реализации, обеспечивающих уменьшение затрат на каждом из этапов. Эффективность ТП, конкретное количество ее этапов и их продолжительность будет зависеть от:

выбранного критерия эффективности ТП (в качестве которого предлагается максимальный уровень надежности при снижении стоимости технологических шагов);

характера разрабатываемой системы (в частности, для управляющих систем характерным является значительное сокращение этапа сопровождения и повышение роли всех предыдущих этапов);

.использования современных методов и средств реализации всех технологических шагов, начиная с внешних спецификаций, позволяющих достигать максимального качества программ при снижении их стоимости;

особенностей объекта управления (ОУ), для которого разрабатывается система. .

1.2.2. Технология проектирования и особенности объектов управления

Используемые в настоящее время технологические принципы проектирования алгоритмов и программ для управляющих вычислительных комплексов являются обобщением опыта разработки систем управления либо уникальными объектами, либо объектами, хотя и составляющими некоторое множество, но эквивалентными по технологии проектирования (отличие от технологии проектирования управляющих алгоритмов для уникальных систем состоит в необходимости тиражирования ПО).

Вместе с тем существуют ОУ, обладающие функциональной однородностью (похожие, подобные объекты). Они не являются ни уникальными, ни эквивалентными, а представляют собой совершенно иной класс объектов, у которых структура едина, функции управления и цели одинаковы, однако объем и содержание различны (например, СУ летательными аппаратами одного и того же типа, железнодорожными станциями).

Пусть B = £Ь.,$Ь_0$,...,Ь„} - множество объектов, обладающих функциональной однородностью, c = {c_1,c_2,...,c ,?} множество признаков, принципиально присущих любому из объектов Ъ-L є B, i=TX F={f_1,f_2,...}; = {c_1,c_2,...,c_m} - множество допустимых функциональных связей между признаками и пусть £(il ,d.?,... ,0V^1,f- характеристическое множество признаков объекта b^, а P = {$p_г$, p_2,..., p^} - множество функционально законченных текстов ПО, удовлетворяющих условию р.~ Ь.. Тогда, если uD.=nD ., 1 =1"71Г, то $p_п$-=$p_п$«,

гии проектирования и разработки ПО для любого из объектов ьГ

При выполнении условия ₁up тексты ПО для любого из объектов b_i находятся в соотношении (vie~(T7k)) (ajefTTk)) (P-^P-,.) и в предельном случае (vi, J є (ITT), l*j) (p.*p.) являются уникальными.

Уникальность текстов ПО предполагает и уникальность технологий его разработки, хотя в ряде случаев очевидность этого и небесспорна. Здесь следует различать понятия:

технологии программирования как описание процесса программирования (своего рода ''know how”);

технологии программирования как самого процесса создания программного обеспечения во времени.

Технология программирования как ''know how" представляет некоторую макротехнологию разработки всех программных продуктов. ТП - "know how" для всех объектов (уникальных, равных, функционально-однородных) должна быть единой. Технология программирования как сам процесс создания ПО -во времени (с учетом методологии) является своего рода реализацией ТП - "know how" во времени и, соответственно, зависит от объектов, для которых разрабатывается программный продукт.

В частности, ТП как сам процесс создания ПО во времени для одинаковых (равных) объектов должен быть единым, для уникальных (различных) - уникальным.

Степень неэквивалентности функционально однородных объектов определяет различия в ПО управляющих ими систем. Для каждого из объектов процедура проектирования управляющих алгоритмов (по крайней мере, ряда этапов) носит уникальный характер и завершается созданием их машинного эквивалента -

уникальных текстов прикладного ПО.

Однако, необходимость прохождения одного и того же трудоемкого пути разработки ПО для объектов, "управляемых по одним и тем же законам, но отличающихся друг от друга количеством входов-выходов, значениями коэффициентов и т.п. в данном случае противоестественна. Это приводит к повторению ряда технологических шагов для каждого отдельного объекта, что является источником ошибок и увеличивает количество итераций (и, естественно, трудоемкость разработки) на этапе тестирования и отладки. Степень похожести, эквивалентности функционально однородных объектов подсказывает возможность реализации таких технологических процессов, в которых большая часть технологических шагов - общая для выделенных объектов. То есть, ТП как сам процесс содания ПО во времени для функционально однородных объектов в целом будет различной (уникальной), однако имеется возможность создания такой единой технологии проектирования для объектов выделенного класса, у которой должно быть минимум повторяемых технологических этапов во времени.

Таким образом функциональная однородность ОУ позволяет построить автоматизированную систему генерации текстов их ПО по описанию конкретного объекта, исключающую этапн проектирования логики управляющих программных модулей, кодирования, тестирования, отладки. Эти этапы разрабатываются один раз для всех объектов данного класса, что позволяет говорить о снижении стоимости разработки ПО в целом для указанных объектов.

Поэтому основной целью исследования является разработка

методики создания ПО для объектов управления подобного типа, позволяющей создавать надежное эффективное ПО с минимальными затратами на его разработку.

Применительно к системам, управляющим функционально однородными объектами, ее следует решать по двум направлениям:

минимизация затрат на отдельных технологических шагах;

увеличение технологических этапов, реализация которых является общей для всего класса выделенных объектов, т.е. свести к минимуму затраты на привязку ПО к конкретному объекту.

Единую технологию разработки ПО и общую структуру программного обеспечения, ориентированную на класс функционально однородных объектов, целесообразно синтезировать на базе принципов функциональной избирательности и избыточности (их содержание будет подробно раскрыто в главе 3).

Все ОУ условно можно разделить на:

уникальные (различные),

равные (одинаковые),

подобные (функционально однородные).

Два объекта управления а и Ь будем считать уникальными (различными), если ими управляют-уникальные (различные) СУ.

Два объекта управления а и Ъ будем считать равными (одинаковыми), если ими управляют равные (одинаковые) СУ. Необходимо учесть, что объекты могут быть не равны в обычном смысле слова, но с точки зрения управляющих ими систем они одинаковы.

ТГрр /4UU пЛт.ртпшо о Jtk м. тл h Лтрттом птгитофт. rfnruwTrwnwQ XXXUIUjJtXV'XXCIWXXJXXS-/ ЧУуЦХХ^^/Уу^ НА ттт.тіп птгпптпг»тгитл—

ми (похожими), если они сочетают в себе черты и уникальных,

и равных объектов, то есть:

количество элементов в объектах а и Ъ может быть различно,

тип содержащихся в объектах элементов может быть различным,

связи между элементами в объектах могут быть различными, однако,.объекты имеют одну область применения (цели одинаковы),

архитектура объектов а и b идентична, т.е. принципы построения связей между элементами в объектах а и b едины,

управление объектами а и б производится по одним общим законам т.е. функции управления одинаковы.

Выделение функционально однородных объектов в отдельный класс дает возможность:

разработать стратегию проектирования ПО в целом,

реализовать такие технологические процессы, в которых большая часть технологических шагов - общая для выделенных объектов,

при проектировании ПО для конкретных ОУ сконцентрировать усилия только на тех его составляющих, которые являются носителями различий между объектами данного класса.

,1.3. ВЫВОДЫ К ГЛАВЕ 1

1. Современные требования к программным продуктам приводят к необходимости измерять и прогнозировать характеристики качества программ и изучать зависимость этих характеристик от различных параметров. Проведена оценка множества характе-

ристик качества ПО и выделены элементарные .характеристики, промежуточные характеристики и характеристики более высоких уровней.

2. Исследована иерархическая структура характеристик, позволяющая выявить взаимосвязи между характеристиками различных уровней и оценить влияние изменения элементарных характеристик на характеристики более высокого уровня.

3. Анализ надежности крупных программных комплексов показал, что надежность ПО оценивается по аналогии с надежностью аппаратных средств, что существенно снижает надежность ПО. Имеются попытки оценить надежность ПО, не проводя аналогии с надежностью аппаратных средств [26,79,88,1261.

4. Предложено множества С всех возможных технологий,' обеспечивающих требуемый уровень надежности, рассматривать как произведение множества А технологических шагов и множества В средств их реализации. Так как выбрать из множества 0 технологию, обеспечивающую минимум прямых и косвенных затрат на создание ПО, невозможно из-за невозможности построения А и В, то предлагается поиск ограниченного подмножества компонентов множеств А и В, обеспечивающих требуемый уровень надежности при минимальных затратах. На этом подмножестве предлагается построить технологию проектирования ПО.

5. В ходе исследований выявлено, что' эффективность технологии проектирования, конкретное количество ее этапов-и их продолжительность зависит от:

выбранного критерия эффективности ТП (в качестве которого предлагается - максимальный уровень надежности при снижении стоимости технологических шагов);

характера разрабатываемой системы (в частности, для СУ характерным является значительное сокращение этапа сопровождения и повышение роли всех предыдущих, этапов);

использования современных методов и средств реализации всех технологических шагов, начиная с внешних- спецификаций;

особенностей объекта управления, для которого разрабатывается система.

6. Даны определения уникальных, равных и подобных объектов управления. Сделан вывод о возможности построения эффективных технологий проектирования программного обеспечения для функционально однородных объектов управления. Используя функциональную однородность объектов управления, предагается построить автоматизированную систему генерации текстов их программного обеспечения по описанию конкретного объекта, исключающую этапы проектирования логики управляющих программных модулей, кодирования, тестирования, отладки. Эти этапы разрабатываются один раз для объектов подобного типа и приводят к снижению стоимости создания ПО, не уменьшая при этом надежности.

7. Дана общая постановка задачи поиска методики разработки ПО для систем, управляющих функционально однородными объектами, решение которой позволит создавать надежное эффективное программное обеспечение с минимальными затратами на его разработку.

2. ГОМОГЕННЫЕ ОБЪЕКТЫ УПРАВЛЕНИЯ

2.1. Определение гомогенности объектов управления

Концептуальной основой построения эффективной с точки зрения соотношения между стоимостью и надежностью технологии программирования для функционально однородных объектов управления является вводимая формальная математическая модель всего класса подобных объектов.

Функционально однородные объекты дальше будем именовать гомогенными (от греческого ''homogenes'' - однородный) [110].

Будем считать объекты а и Ъ гомогенными, если:

совокупности элементов а и Ъ образуют кортежи над одним и тем же множеством М,то есть:

$a(a) = \langle a_1, a_2, \ldots, a_K \rangle \in A;$

$h(p) = \langle P|,\hat{}2»\bullet\bullet\blacksquare\rangle P_T \rangle \in B;$

v a_Lе м, 1 = 1,... ,к ;

v р е м, 3 = 1,...,т ,

и

где а(а), Ъ(р) - кортеж а модели объектов а и кортеж р модели объектов Ъ соответственно,

А - множество кортежей длиной к над множеством М, мощность множества А - $п^к$;

В - множество кортежей длиной m над множеством М, мощность множества В - ^1д=$п^т$;

п - мощность множества М;

объекты а и Ъ имеют одну область применения;

архитектура объектов а и Ъ идентична, то есть принципы построения связей между элементами в объекте а и элементами

в объекте Ъ едины;

управление объектами а и Ъ производится по одним общим законам.

Объекты а и Ъ считаются равными (одинаковыми) только тогда, когда кортежи аир равны и соотношения между компонентами кортежа а совпадают с соотношениями между компонентами кортежа р.

Пусть а = <a_1,a_2,...,$a_к$ >,

р = <p_t,p_2,...,p > - два кортежа, представляющие собой формальные модели некоторых объектов а и Ъ .

Поставим во взаимно однозначное соответствие компонентам кортежа а некоторое множество R, а компонентам кортежа р-множество S, то есть:

Л = ,ot_2 , . . . ,CC^.z^ < > R = {г^,$г_2$,..., $г_к$},

P = <p^,P_2,...,p^ > < ▶ S = {s_{at},s_2,..., Sj^},

где a_L*→ r_L, P-J«→ S-J, 1= GG > Gm.

Определим множество Z как Z=RuS= {z_1,z_2,...,z }, где m+k > n > max { m,k),

а множество 7"как = Z^2= Z*Z ={<z_L, z^>}, 1,J= Gn, .

Тогда ТГ =R^2= {<r_L,r.j>}, i,>G&,

и 'S"=S^2= {<s_t,8j>}, l,j=Gm, ixj.

Так как R s Z и S s Z,то H,*S <= Tı.

Теперь совокупность соотношений между компонентами кортежа а можно определить как некоторое множество F а для кортежа р - Т = где F и Т - отношения на множествах R и S. Если <r_L,Pj> F(<S_L,S^> е Т), то г^Е г^ (S_L Т S_J), что читается как "r_L(s_L)" находится в отношении E(T) с , а само выражение "rj гу' ("sj зу') и будем

называть соотношением [125].

Так как F s H, T s S, H, S s Z, то и F,T s Z. Тогда соотношения между компонентами кортежа а и компонентами кортежа р будут равны, если F=T.

Так как множества R и S определены в виде взаимно однозначного соответствия (1), то. и будут справедливы соотношения a.F a_4 и р.Т р ..

-Ь J . J- J

Так как кортежи аир являются моделями объектов а и Ь, то необходимым, но недостаточным условием их равенства является равенство кортежей а и р.Отсюда еледует,что R=S и, соответственно, Z= RuS=R=S, R= S= Z.

Так как соотношения a.Fa-и р.тр. по {1,2,...,к},

-L J X J

принадлежащему одному и тому же множеству Z, то достаточным условием равенства объектов а и b будет равенство отношений ? и Т на множестве Z.

Конструктивные выводы из приведенного доказательства могут быть сформулированы в виде следующих формальных условий равенства объектов а и Ъ:

<a^ ,<^2 >•••»&£ >=<p^ , P2 ,. •., p^. >,

Z? i,J=17k, 1^3,

р.ТрлЕ Z? 1,>ЇД,

F = T,

Z = RuS,

R = {rp, S = {sp, r_i<–► a_A, $8_±$<–> p_x ,l=l,k,

R = S,

F,T s Z - отношения на множестве Z.

Объекты а и b различны тогда» когда не выполняется хотя бы одно условие помощности

Понятие гомогенности как абстрактной характеристики объекта управления практически отображается в ПО СУ подобными объектами, поэтому концепция гомогенности применима и к нему. В дальнейшем понятие гомогенности будет применяться без особых оговорок и к ПО СУ объектами, относящимися к классу гомогенных, и к отдельным его частям, образованным при декомпозиции ПО на составляющие.

Таким образом, выделение гомогенных объектов в отдельный класс дает возможность:

разработать концепцию проектирования программного обеспечения для выделенного классса объектов в целом;

реализовать такие технологические процессы, в которых большая часть технологических шагов - общая для выделенных объектов. Это позволяет, как показано в 1.2.2., построить эффективную с точки зрения надежности и стоимости системы.

2.2. Разработка технологии программирования систем управления гомогенными объектами

2.2.1. Основные определения

Для решения поставленной в работе задачи разработки концепции технологии проектирования программного обеспечения систем управления гомогенными объектами, позволяющей создавать надежное ПО с минимальными затратами на его разработку, необходимо выяснить: .

содержание технологических этапов, реализация которых является общей для гомогенных ОУ;

какие составляющие проектируемого ПО для конкретного объекта являются носителями различий между объектами данного класса.

Описанную в 1.2. этапность проектирования программных систем можно рассматривать как макротехнологию разработки, не затрагивающую технологических особенностей отдельных процессов. Эта макротехнология включает в себя все процессы, которые должны выполняться при разработке независимо от частных их особенностей.

Весь этап проектирования программ от зарождения идеи до получения готового текста, подлежащего тестированию и отладке, включает в себя несколько различных процессов. При хорошей организации проектирования они ярко выражены, что позволяет установить контрольные сроки, выбрать методологию и по завершении каждого процесса проверить качество полученных результатов. На рис. 9 изображена модель процессов проектирования сложной системы.

На первом шаге составляется перечень требований, которые могут разрабатываться либо непосредственно организацией-пользователем (заказчиком), а разработчик программного обеспечения является ее субподрядчиком (проект, управляемый пользователем), либо совместными усилиями разработчика и заказчика (проект, контролируемый пользователем), либо только разработчиком (проект, независимый от пользователя).

Следующий шаг - это постановка целей - задач, которые ставятся перед окончательным результатом и самим проектом.

А Этапы проектирования и разработки программного ^6
обеспечения.

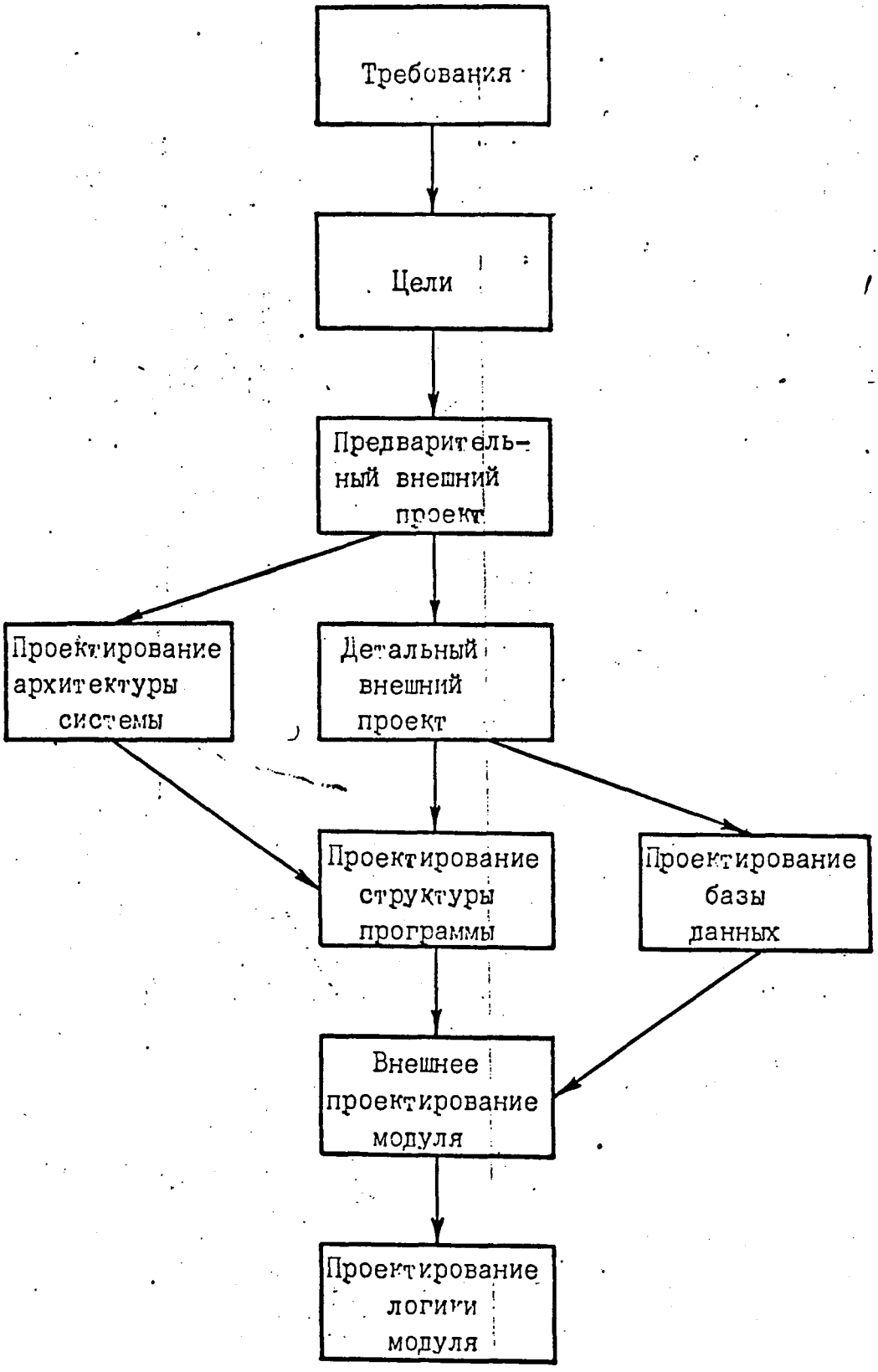


Рис. 9 ■

При правильной постановке целей не делается никаких предположений о конкретной реализации, но указывается, каким образом на последующих этапах проектирования следует принимать компромиссные решения. Должны быть поставлены цели двух типов: цели продукта (готовой программы) и цели проекта. Первые - это цели, с точки зрения пользователя; вторые - все то, что относится только к самому процессу разработки (график, стоимость, критерии для оценки готовности и тому побное) и не проявляется в окончательном результате работы.

Затем выполняется предварительный внешний проект, в котором рассматривается взаимодействие с пользователем, но не затрагиваются многие его детали, например форматы ввода-вывода. В процессе детального внешнего проектирования завершается определение взаимодействия с пользователем, описываются его мельчайшие подробности.

Внешнее проектирование - это процесс ожидаемого поведения разрабатываемого программного изделия с точки зрения внешнего, по отношению к нему, наблюдателя. Цель этого процесса - проектирование внешних взаимодействий будущей программы с пользователем без конкретизации ее внутреннего устройства. Внешний проект выражается в виде внешних спецификаций.

В процессе разработки архитектуры системы выполняется разложение ее на множество программ, подсистем или компонент или определяются сопряжения между ними. Детальный внешний проект и проектирование архитектуры системы предшествуют проектированию структуры программы, в которой разрабатываются модули,, их сопряжения и взаимосвязи для каждой программы

или подсистемы.

Следующий процесс - внешнее проектирование модуля - это точное определение всех сопряжений модуля, которое состоит в определении всех его внешних характеристик. Внешне этот процесс выражается в виде внешних спецификаций модуля, которые содержат все сведения, необходимые вызывающим модулям. В частности, внешние спецификации не должны содержать никакой информации о логике модуля или внутреннем представлении данных.

Последний шаг - проектирование логики модуля - состоит в разработке внешней логики каждой системы и выражении ее текстом конкретной программы.

Этап проектирования базы данных может отсутствовать (как процесс проектирования архитектуры системы), но если он есть, то состоит в определении всех внешних, по отношению к программной системе структур данных.

Анализ разрабатываемого ПО СУ гомогенными объектами (в частности - микропроцессорной централизации) дает основание предполагать, что от привязки к конкретному объекту зависят в большей степени не сами программы, а их информационные массивы. Поэтому можно говорить о возрастающей роли этапа проектирования базы данных и о его необходимости для гомогенных объектов.

Изображенная на рис. 9 схема процессов в некоторой степени упрощена. При работе над реальным проектом последовательность их более сложная. Между каждым из процессов существуют обратные связи, проявляющиеся в результате возможного обнаружения ошибок на каком-нибудь из этапов. Так, если при

проектировании структуры программы выявлены погрешности в формулировке целей, то следует немедленно вернуться и исправить их. Или на этом же этапе может быть обнаружено, что указанная во внешних спецификациях функция неосуществима или обойдется слишком дорого.

Рассмотренная модель процессов не зависит от методологии программирования. Все указанные в ней действия должны выполняться в любой разработке независимо от того, какой язык программирования выбран, какие принципы использовались при проектировании логики модуля и его кодирования.

Для ответа на вопрос - какие составляющие проектируемого ПО являются носителями различий между объектами данного класса - целесообразным представляется уточнить понятия ''программное обеспечение ЭВМ'', ''программное обеспечение сис-
темы”, ''системное программное обеспечение системы", "прикладное программное обеспечение системы", "программа", ее составляющие.

2.2.2. Программное обеспечение системы управления

Рассмотрим вычислительную систему самого общего назначения на базе универсальных ЭВМ.'

Выделим две наиболее популярные формы- их использования: решение отдельных научных, инженерных, экономических и прочих задач;

решение комплекса взаимосвязанных задач, подчиненных единой цели и образующих систему (например, автоматизирован-

Технологии взаимодействия человека с ЭВМ в каждом из вариантов и режимы использования вычислительных установок принципиально не отличаются. В каждом из вариантов между двумя противоположными во времени актами - зарождением идеи и ее реализацией с помощью ЭВМ - выполняются ряд шагов. Весь процесс в целом носит итеративный характер и распадается на две самостоятельные проблемы:

проектирование логики решения задачи и получение машинной программы (в дальнейшем будем называть ее прикладной);

получение результатов на допустимых наборах исходных данных и их интерпретация человеком.

Различия между упомянутыми формами использования ЭВМ состоят в характере обращения к услугам ЭВМ: в первом случае он спорадический (от случая к случаю), во втором - систематический.

На этапе проектирования логики и получения машинной программы кроме чисто аппаратных средств ЭВМ используются и специальные программы, которые можно рассматривать как программное продолжение ЭВМ. Комплекс таких программ принято называть программным обеспечением ЭВМ.

Разработанная на некотором входном языке (как правило, отличном от внутреннего машинного языка и недоступного для непосредственной аппаратной интерпретации) прикладная программа подлежит вводу в ЭВМ с последующим анализом синтаксической корректности текста. На этом шаге используются такие компоненты программного обеспечения ЭВМ как программы ввода-вывода (драйверы ввода-вывода), программы синтаксического контроля текста и вывода сообщений о результатах конт-

роля. Если результаты контроля негативны, то производится исправление ошибок и вновь повторяется предыдущий шаг. Эти процедуры повторяются до тех пор, пока не будет получено сообщение, что синтаксических ошибок больше не обнаружено.

После этого начинается шаг перевода исходной программы на внутренний машинный язык. В этом случае используются такие компоненты программного обеспечения как трансляторы, редакторы и загрузчики.

На следующем шаге производится исполнение разрабатываемой программы на некоторых контрольных (тестовых) наборах исходных данных с целью обнаружения возможных- ошибок в логике. Этот шаг в разработке принято называть тестированием. В результате тестирования устанавливается только факт неправильной работы прикладной программы, то есть факт наличия в ней ошибки. Сама же физическая природа ошибки остается неизвестной. И целью следующего шага - отладки - является установление места и сути ошибки. Очень часто проще обнаружить факт существования ошибки, чем установить ее физическую природу. Если ошибка не только обнаружена, но и исправлена, вновь повторяется шаг тестирования. Оба шага повторяются до тех пор, пока возможности теста не будут использованы полностью и не будет сделано заключение, что обнаружить ошибки больше не удается. Это обстоятельство дает основание разработчику прикладной программы утверждать, что он достиг конечной цели - разработал программу решения поставленной задачи.

На шаге отладки могут использоваться такие компоненты программного обеспечения ЭВМ, как специальные отладочные

программы (программы' трассировки, печати дампов и тому подобное).

Прикладная программа является самоцелью только для ее автора. Конечная же цель - это получение с ее помощью для различных допустимых наборов данных результатов вычислений в соответствии с логикой программы. Эта конечная цель достигается на втором этапе, когда в ЭВМ вводятся данные только для прикладной программы и интерпретирующая ее аппаратура ЭВМ выдает в требуемой форме результаты вычислений.

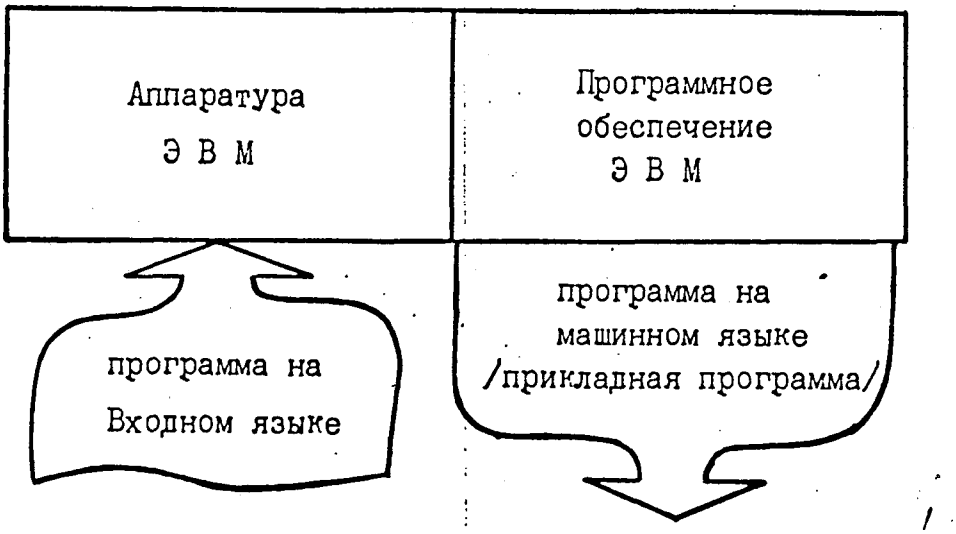
Взаимный статус программного обеспечения ЭВМ и прикладной программы на обоих этапах принципиально различны.

На первом этапе программное обеспечение ЭВМ, по отношению к прикладной программе является, наряду с аппаратурой, инструментом ее создания (рис. 10).

На втором этапе прикладная программа становится в один ряд с компонентами программного обеспечения ЭВМ и вместе с ним и аппаратурой становится инструментом решения некоторой частной узкоспециализированной задачи (рис. И).

Пусть необходимо время от времени производить расчеты траектории полета на Луну в зависимости от времени старта летательного аппарата и пусть программа решения этой задачи разработана и находится в памяти ЭВМ. Прикладная программа погружена в среду, обеспечиваемую аппаратурой ЭВМ и ее программным обеспечением. Но тогда мы вправе считать, что поставленная задача решается системой, состоящей из ЭВМ (аппаратные средства) и некоторого комплекса программ (программные средства). Условно назовем такую систему ’’Система расчета траектории”.

Средства разработки прикладной программы



Рис, 10

Средства решения прикладной задачи

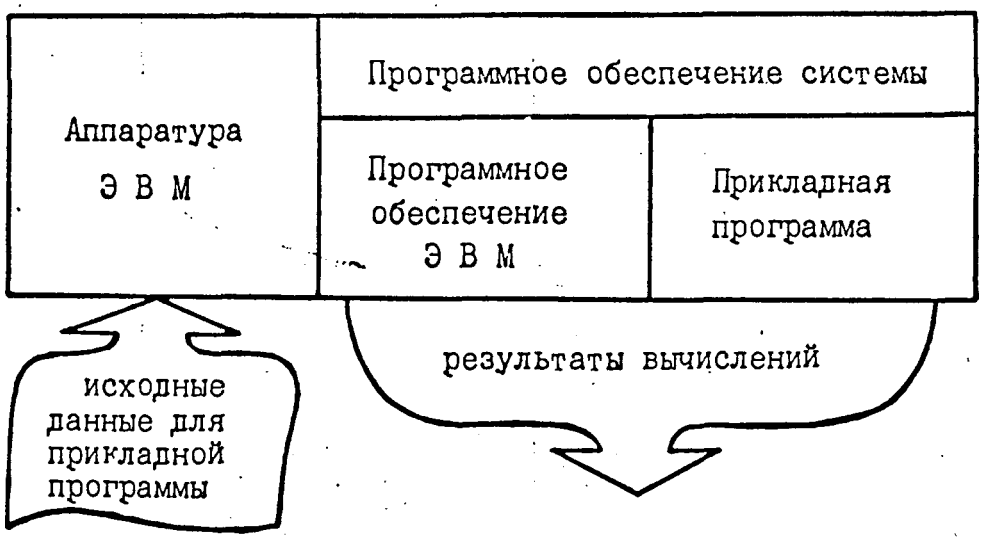


Рис.11

Логические возможности такой системы реализуются аппаратурой ЭВМ и рядом программ, которые и назовем программным обеспечением ''системы расчета траектории".

Здесь уместно обратить внимание на различие понятий ''программное обеспечение ЭВМ" и ''программное обеспечение системы". Программное обеспечение ЭВМ - это комплекс программ, осуществляющих наиболее общие функции в процессе решения любой прикладной задачи и являющийся, по сути, программным продолжением аппаратных средств. Программное обеспечение ''системы..." состоит из некоторых компонентов программного обеспечения ЭВМ и прикладной программы (или прикладных программ).

Так как функциональная ориентация каждой из этих составляющих резко отличается друг от друга (в одном случае наиболее общие функции, в другом - специализированные, частные), то только в этом случае и появляется необходимость в введении понятий системного программного обеспечения ''системы..." и прикладного программного обеспечения "системы...".

Эти различия в понятиях основаны на том обстоятельстве, что у аппаратуры ЭВМ и ее программного обеспечения очень ограничены области определений и области значений, то есть ограничены формы исходных данных - только тексты на входных языках и форма выходных данных - текст на внутреннем машинном языке.

Расширяя же функциональные возможности ЭВМ за счет прикладной программы, мы создаем новую виртуальную машину, имеющую тот же состав аппаратуры, но другое программное обеспечение.

Целесообразное поведение любого объекта достигается воздействием на него некоторой метасистемой, которая по отношению к этому объекту является управляющей, и замкнутый контур управления может быть представлен как совокупность двух составляющих: объекта управления и системы управления, соединенных информационными связями (рис. 12).

С точки зрения внешнего наблюдателя способы достижения тех или иных функций системой управления неразличимы. Вместе с тем эти функции могут быть реализованы как логически, так и конструктивно самыми различными способами. Различия между вариантами могут быть абстрактно оценены по таким критериям как технологичность, доступность для реализации, эргономичность, ремонтопригодность и ряду других [85].

Допустим, что оцениваются два варианта системы управления, эквивалентные в своей конечной фазе: вариант реализации всех функций управления по жесткой логике ' (исключительно аппаратно) и вариант, допускающий реализацию некоторых функций по гибкой логике (то есть по программе, отражающей специализированную логику, но интерпретируемую по логике, не зависящей от конечных целей управления).

Пусть по ряду соображений предпочтение отдано второму варианту. Очевидно, что каким бы мощным не было множество функций системы управления, реализуемое по программе (мягкий то вар - soft ware), оно не может содержать всех функций, подлежащих реализации. В конечном счете, часть из них должна выполняться аппаратно (твердый товар - hard ware) (в пределе - машина Тьюринга).

Замкнутый контур управления

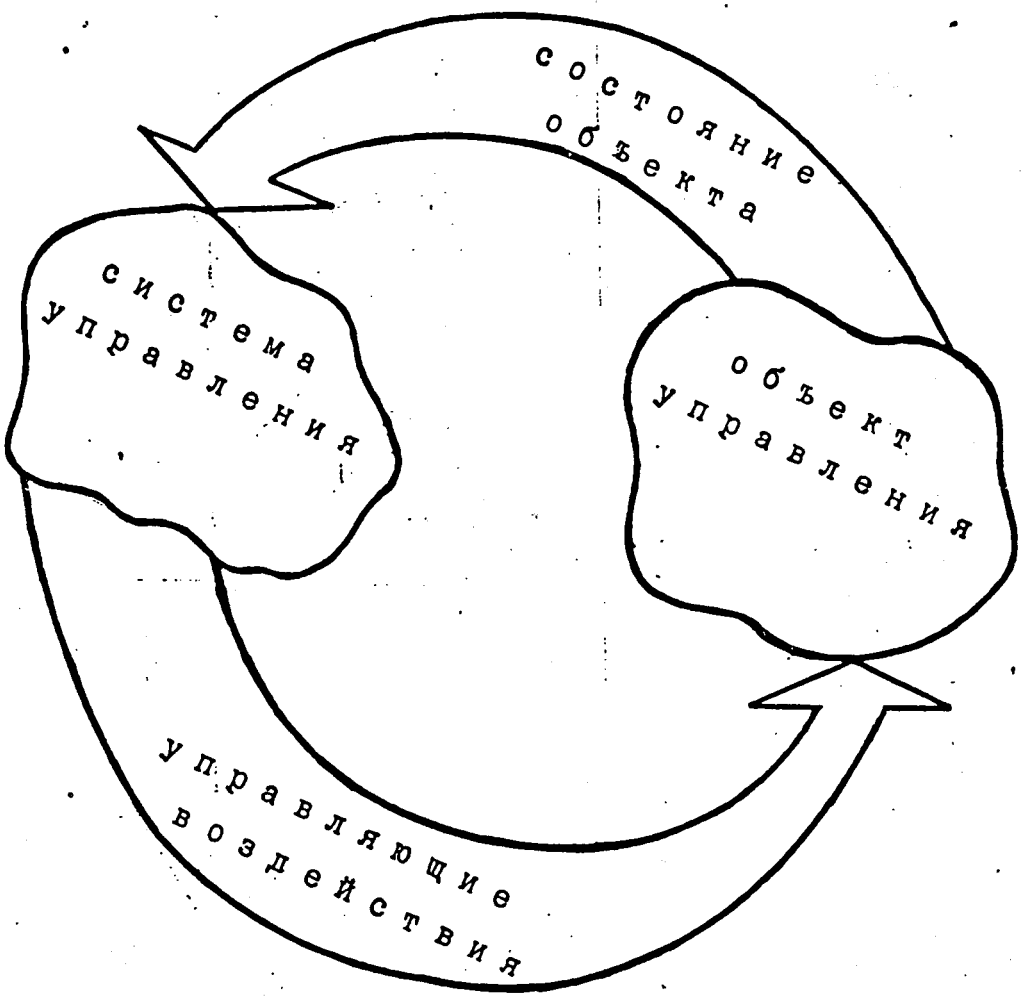


Рис. 12

Таким образом, в системе управления, построенной на базе программируемых вычислительных средств, часть функций выполняется аппаратно, а часть - программно. При этом одни функции могут представлять самостоятельную ценность, а другие - быть инструментом реализации более сложных функций.

Таким образом, к программному обеспечению системы управления следует отнести (рис. 13) программное расширение функций вычислительной машины: функции, определяемые поставленными задачами; функции, определяемые характеристиками объекта управления.

Между функциями, реализуемыми аппаратно, и функциями, реализуемыми программно, существует граница (интерфейс), которая определяет объем программной составляющей. Эту составляющую и будем называть программным обеспечением системы управления. Положение границы зависит от развитости системы аппаратной интерпретации. Для вычислительных машин с развитой системой аппаратной интерпретации граница сдвинута вправо, а для машин с примитивными системами аппаратной интерпретации граница сдвинута влево.

Программная составляющая представляет собой некоторое подмножество функций системы управления, каждая из которых реализуется достаточно независимым программным блоком (модулем). Каждый программный модуль, его логика и информационные связи имеют различную степень зависимости от целей управления и характеристик объекта управления.

Распределение функций между, аппаратурой и программным обеспечением в системе управления

система управления

Функции, реализуемые . аппаратно’	Программное расширение функций вычислительной системы	Функции, определяемые характеристиками задач	I 1 ■Функции, непосредственно определяемые характеристиками объекта управления

аппаратная часть

программное обеспечение

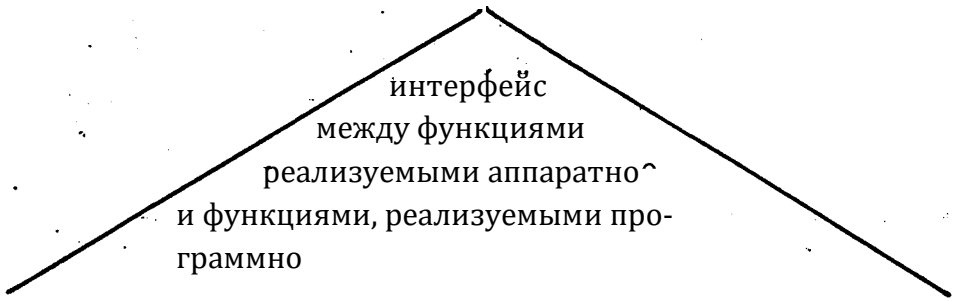


Рис.13

ел
оо

Если расположить модули по возрастанию их зависимости от особенностей объекта управления, то левую границу этого спектра составят программные средства, абсолютно не зависящие не только от целей управления и характеристик объекта управления, но даже от самых общих характеристик, решаемых в процессе управления задач. Такие модули можно считать программным продолжением самой вычислительной машины. Ими могут быть программы обмена данными с внешними устройствами (драйверы ввода-вывода), обмена данными между каналами для резервированных систем, программы мажоритирования, программные средства диагностики состояния аппаратуры, программные средства адаптации структуры системы управления и тому подобное .

На противоположной, правой, границе спектра располагаются модули, реализующие основные функции управления.

Если мы назовем представителей левой границы системными модулями, а правой - прикладными, то по мере изменения зависимости от особенностей процесса управления прикладное и системное программные обеспечения где-то смыкаются. В отличие от интерфейса между аппаратными и программными функциями граница между прикладными и системными модулями размыта и четко установить ее трудно.

В некоторых случаях целесообразным оказывается выделение промежуточной группы модулей (средней части спектра) в отдельный класс системно-прикладных модулей. Эти модули не реализуют непосредственно функций управления, но поддерживают их, обеспечивают необходимую операционную обстановку, в которую погружаются прикладные модули. В то же время они и

не являются программным продолжением машины, хотя и опираются на ее аппаратные средства (например, систему прерываний) и на чисто системные программные модули (например, драйверы ввода-вывода).

Под прикладным программным обеспечением целесообразно понимать программные средства, реализующие основные функции управления. На рис. 13 это - левый конец спектра.

Структурно прикладное ПО представляет собой совокупность модулей. Все модули имеют различные количественные запросы ресурсов (памяти и процессорного времени), но на уровне организации общего вычислительного процесса порождают задачи двух типов: циклические и разовые. Эти характеристики и влияют на промежуточную часть спектра программного обеспечения, отнесенную также к системному ПО управляющей системы.

Таким образом, при формулировке принципов разработки ТП для гомогенных объектов управления будем придерживаться следующей терминологии: программное обеспечение системы управления; внутреннее ПО (тексты на внутреннем языке ЭВМ); ПО объекта управления (часть внутреннего ПО, поставляемого с ЭВМ на конкретный объект управления); системное ПО СУ (часть внутреннего ПО, общая для всех объектов данного класса); прикладное ПО объекта управления (уникальная составляющая ПО ЭВМ).

2.2.3. Информационный подход к проектированию программного обеспечения систем управления гомогенными объектами

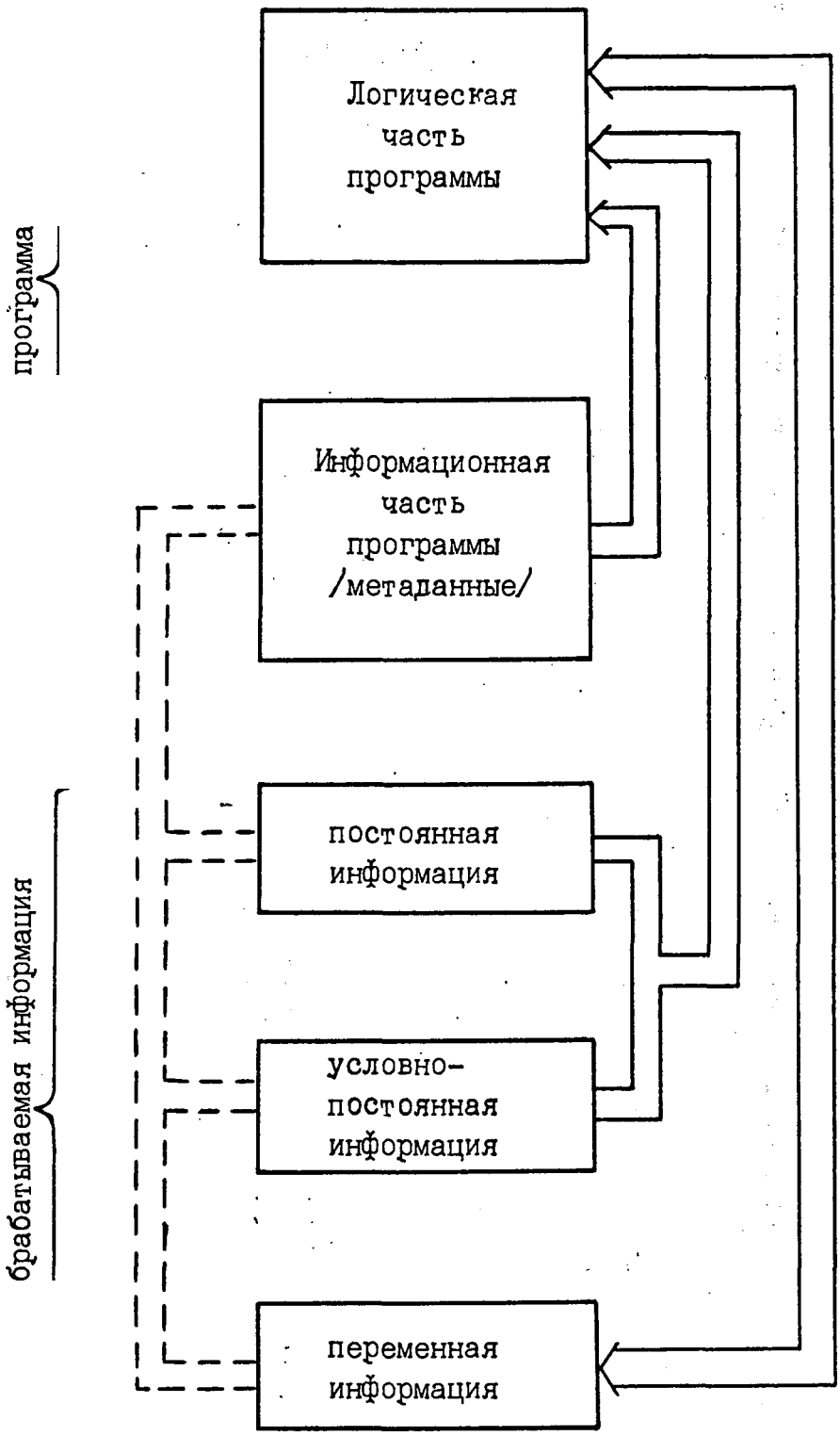
Как следует из 2.2.2., выходным продуктом прикладного программирования для гомогенных ОУ являются не программы, а сопровождающие их информационные массивы, статические для одного объекта, динамические для других объектов. То есть прикладное программирование будет представлять в качестве конечного продукта не программы, а уникальные наборы данных. Поэтому и необходима унификация информационной базы.

При формулировании принципов, составляющих единую для гомогенных объектов управления технологию, под программой будем понимать совокупность команд (логическая часть программы), интерпретируемых процессором управляющего вычислительного комплекса, и сопровождающие их константы (информационная часть программы). Программу можно представить графически в виде, изображенном на рис. 14.

Вся информационная базауправляющей системы по степени ее постоянности (неизменности) делится на следующие группы: постоянная информация (нормативно-справочная), условно-постоянная,
переменная.

Постоянная информация -данные, общие для любого объекта управления, не зависящие ни от конкретного объекта, ни от процессов, протекающих на нем во время управления. Таковыми являются, например, коды директив, их числовые эквиваленты, тексты сообщений.

Графическое представление программы



Рис»14

Условно-постоянная информация зависит от конкретного объекта, но не зависит от процессов управления на нем. Ее можно разделить на две подгруппы!

метаданные (данные о данных),.

собственно условно-постоянную информацию - информацию, подлежащую обработке.

Переменная информация характерна для конкретного объекта управления, зависит от процессов управления и изменяется в реальном времени, например! тексты вводимых директив, очередь системных .сообщений.

2.2.4. Введение общих принципов программирования модулей

При решении сложных задач управления резко повышается объем и сложность программного обеспечения. Разработка таких сложных, программных проектов без соответствующей методики проектирования эффективной быть не может. Прежде всего, такая методика дожна предусматривать возможность разработки ПО~ как единого концептуального целостного программного изделия коллективом разработчиков.

Программное обеспечение как изделие еще редко упоминается в литературе. Даже сегодня число людей понимающих, что ПО может быть самостоятельным изделием, к сожалению, невелико. Требуются определенные усилия, чтобы убедить, что объекты, подобные программному обеспечению, которые представляются неосязаемыми, в действительности оказываются реальными изделиями.

Тип программных средств, для которых в большей мере при-

годна концепция программного изделия,- это универсальное программное обеспечение, которое должно многократно копироваться, устанавливаться во многих местах, по разному применяться [15]. Это продукт тщательного планирования, сопровождаемый четкой документацией, прошедший все необходимые испытания, описанный в соответствующих публикациях, обеспеченный обученным персоналом, обслуживаемый и контролируемый поставщиком.

Типичным примером подобного ПО является большинство системных программных средств, поставляемых отдельно от интерпретирующей аппаратуры.

При этом следует понимать, что отдельная программа или совокупность программ и программное изделие - далеко не одно и то же. Непонимание или недооценка концепции программного изделия проявляется прежде всего в отсутствии:

системного подхода к проектированию ПО;

тщательного предварительного анализа условий применения;

достаточной обоснованности проектных решений;

согласованности внешних спецификаций;

четкого планирования и управления проектированием;

надлежащего решения кадровых вопросов.

В результате, программное изделие уступает в качественном смысле аппаратному изделию.

Известный американский специалист в области .проектирования и реализации систем программного обеспечения Р.Гантер [17] выделяет три причины непонимания концепции программного изделия. Первая состоит в том, что программное обеспечение не имеет конкретного физического воплощения: его нельзя по-

трогать, понюхать, взвесить. Оно представляет нечто материальное. Вторая причина заключаектся в том, что,программирование (может быть в силу первой причины) кажется слишком простым занятием. И, наконец, третья причина состоит в том, что большинство разработчиков прогаммного обеспечения или программисты-математики, или инженеры по образованию, не имеющие опыта инженерной работы и, поэтому, мало знакомы с концепцией изделия.

Поставленная цель в настоящее время может быть решена в соответствии с технологией модульного проектирования сверху вниз. Среди специалистов, дающих определение модулю, единого подхода в трактовке нет. Наиболее точным, на наш взгляд, следует признать следующее определение модуля С123J:

модуль - идентифицируемая часть программы, удовлетворяющая определению программмы и обладающая следующими свойствами:

модуль имеет точно поименованные точку входа и точку выхода;

модуль можно вызвать из другого модуля по имени;

модуль может вызывать другие модули по имени;

модуль должен возвращать управление тому модулю, который его вызвал.

Такое определение модуля дает основание, определив функции модуля и способы его взаимодействия с другими модулями (то есть определив информационную среду, в которую модуль погружен), рассматривать его как самостоятельную программу и применять к нему принципы модульного программирования.

Модульное программирование характеризуется следующими

преимуществами:

1. Большую программу могут одновременно писать несколько исполнителей - это сократит время разработки.

2. Можно составить библиотеки наиболее употребительных программ.

3. Становится проще процедура загрузки в оперативную память большой программы, требующей сегментации.

4. Появляется много естественных контрольных точек для наблюдения за разработкой проекта.

5. Обеспечивается более полное тестирование.

6. Проще проектирование и последующие изменения программы при сопровождении.

Три последних пункта особенно важны, так как стоимость сопровождения программ составляет значительную часть общих расходов на программное обеспечение.

Хотя модульное программирование имеет и некоторые недостатки:

1. Несколько увеличивается время исполнения программы.

2. Может возрасти размер требуемой памяти.

3. Может увеличиться время компиляции и загрузки.

4. Проблемы 'организации межмодульного взаимодействия могут оказаться достаточно сложными, но имеющиеся преимущества многократно перекрывают их.

Суть модульного программирования состоит в том, что все программное обеспечение разбивается на самостоятельные части (модули),имеющие минимальные связи друг с другом. То есть модули должны быть в максимальной степени независимы и удовлетворять определению программы. Вместе с тем отдельно взя-

тый модуль, рассматриваемый как самостоятельная программа, работает в среде, создаваемой некоторым другим модулем. Таким образом, между модулями существует отношение подчиненности и в целом модульная структура ПО ОУ может быть представлена в виде нисходящей иерархической структуры. Полная структура может быть известна, как это не парадоксально, в конце проектирования, но начальная его фаза, (нулевое приближение) может быть разработана только после таких этапов как формулировка целей продукта (целей ПО СУ) и внешнего проектирования (разработки внешних спецификаций).

Модули должны соответствовать следующим требованиям:

обладать минимальной связностью;

характеризоваться максимальной прочностью.

Модули, составляющие программу, не только выполняют функции, для реализации которых они предназначены, но и образуют определенные связи между собой. Хорошая модульная программа должна обладать минимальной связностью модулей, а ее модули - максимальной прочностью. Степень связности модулей определяется формой передачи данных между модулями и назначением передаваемых данных. Для обеспечения минимальной связности модулей по данным необходимо ликвидировать жестку'ю привязку разрабатываемой системы к конкретному объекту, то есть унифицировать информационную модель.

Исходя из введенных в 2.2.2. определений программы и терминологических определений, представляющих необходимую классификацию ПО, сформулируем основные принципы единой ТП для проектирования и создания прикладного (характерного для конкретного ОУ) ПО:

1. Все управляющие алгоритмы при проектировании прикладного ПО строятся таким образом, что

логическая часть программы и структура ее информационной составляющей для всех объектов данного класса едины;

все частные особенности отдельных объектов управления определяют только объем и содержание (но не структуру) информационной части программы.

2. Использование методов минимизации условно-постоянной информации.

,3. Прикладное программирование сводится к описанию особенностей конкретного ОУ.

4. Информационная составляющая прикладного ПО генерируется автоматически..

Для реализации указанных принципов необходимо разработать соответствующую программную и лингвистическую поддержку, которые будут описаны в 3-ей и 4-ой главах.

Перечисленные принципы неравнозначны по своему значению в процессе проектирования и создания прикладного ПО. Наибольший удельный вес занимает минимизация условно-постоянной информации. Сведение ее к минимуму дает возможность считать большую часть информации постоянной. При этом используются следующие методы минимизации:

применение современных методов проектирования на модульном и кодовом уровнях. В частности, на стадии проектирования архитектуры целесообразным представляется использование из известных методологий, ориентированных на обработку, методо- ПГ) РТГГЛ МҐМПГ irt.TJr>T>n ТТПГіТттіЛЛЛТЇПТІСПТГЛ'СТ Q W5 momn rrrs nrjpijffr pTyr/ptT—

ровании концептуальных баз данных. Метод, используемый на кодовом уровне проектирования, известен как структурное программирование. Практика показала [13,115] что структурное программирование само по себе не очень эффективно при проектировании больших систем. Для достижения максимальной надежности и снижения стоимости нужно объединить приемы структурного программирования с методологией проектирования архитектуры, включая бригаду главного программиста, проектирование сверху-вниз, библиотеки, поддерживающие процесс развития проекта и тому подобное;

создание единой структуры информации;

применение на кодовом уровне косвенной адресации для реализации понятия.метаданных (данных о данных);

ввод ограничителей на длины всех массивов либо счетчиков для ограничения на длину массивов для настройки программ на размеры информационных массивов.

2.3. ВЫВОДЫ ПО 2-ОИ ГЛАВЕ

1 .Концептуальной основой построения эффективной с точки зрения соотношения между стоимостью и надежностью технологии программирования для функционально однородных объектов управления является вводимая в главе формальная математическая модель всего класса подобных объектов.

Функционально однородные объекты названы гомогенными (от греческого "homogenes" - однородный).

Выделение гомогенных объектов в отдельный класс дает возможность:

разработать концепцию проектирования программного обеспечения в целом;

реализовать такие технологические процессы, в которых большая часть технологических шагов - общая для выделенных объектов.

2 . При решении сложных задач управления резко повышается объем и сложность программного обеспечения. Разработка таких сложных программных проектов без соответствующей методики проектирования эффективной быть не может. В работе предложена модель проектирования и разработки ПО гомогенными объектами как единого концептуального целостного программного изделия. Расширены понятия программное обеспечение ЭВМ, программное обеспечение системы, системное программное обеспечение системы, прикладное программное обеспечение системы, программное обеспечение объекта управления, прикладное программное обеспечение объекта управления, программа, ее составляющие .

3 .Разработана концепция проектирования программного обеспечения управляющих систем гомогенными объектами,предусматривающая возможность разработки ПО как единого целостного программного изделия и включающая:

идентификацию объектов управления по степени гомогенности;

использование принципов разработки единой технологии проектирования для гомогенных ОУ;

применение методов минимизации условно-постоянной информационной составляющей и сведение ее к постоянной составляющей.

4 . Разработана технология проектирования СУ гомогенными объектами, основанная на следующих принципах:

все управляющие алгоритмы при проектировании прикладного ПО строятся таким образом, что

логическая часть программы ***и*** структура ее информационной составляющей для всех объектов данного класса едины;

все частные особенности отдельных объектов управления определяют только объем и содержание (но не структуру) информационной части программы;

прикладное программирование сводится к описанию особенностей конкретного ОУ;

информационная составляющая прикладного ПО генерируется автоматически.

5 .Предложена концепция минимизации условно-постоянной информации, в которой

предлагается использовать современные методологии проектирования на модульном и кодовом уровнях. В частности, для достижения максимальной надежности и снижения стоимости объединены приемы структурного программирования с методологией проектирования архитектуры, включая бригаду главного программиста, проектирование сверху-вниз, библиотеки, поддерживающие процесс развития проекта и тому подобное;

предусмотрено создание и последующее расширение абсолютной библиотеки программных модулей, из которой можно генерировать тексты для конкретного гомогенного объекта;

предусмотрено создание единой структуры информации.Для этого - предлагается использовать на кодовом уровне понятия метаданных (данных о данных), >. применяя косвенную адресацию.

Ввести ограничители на длины всех информационных массивов либо счетчики для ограничения на их длину, что позволит настраивать программы на размеры информационных массивов.

3. ГЕНЕРАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМ УПРАВЛЕНИЯ ГОМОГЕННЫМИ ОБЪЕКТАМИ

3.1. Принципы функциональной избыточности и функциональной избирательности

В общей технологии создания микропроцессорной системы управления проектирование конкретной конфигурации программного обеспечения становится основным по трудоемкости и длительности этапом. Сокращение сроков разработки как специального программного обеспечения, осуществляющего основные целевые функции управления, так и средств автоматизации программирования, становится реальным, если программное обеспечение ориентировать не на конкретный объект управления, а на некоторый класс объектов.

Вместе с тем такой подход к проектированию программного обеспечения порождает две противоречивые цели: с одной стороны, такая система должна обладать определенной универсальностью с тем, чтобы охватить все возможные функциональные признаки процесса управления любого из объектов данного класса, а с другой - должна иметь минимальные расходы ресурсов микро-ЭВМ, таких как объемы оперативной и постоянной памяти и времени исполнения.

Необходимость удовлетворения двум противоречивым целям приводит к двум различным, но не независимым проблемам. Первая из них состоит в определении функциональных возможностей программного обеспечения, а вторая - в определении программных блоков, которые позволяют достичь эффективной реализации любого из функциональных признаков.

Пусть некоторый объект управления данного класса имеет

множество $A_i = \{a_1, a_2, \ldots, a_n)$ функциональныхпризнаков, определяющих состав и структуру его программного обеспечения.

Тогда общее множество функциональных признаков, характеризующее весь класс объектов управления, можно представить в виде:

$$A = \bigcup_i A., \qquad (2)$$

а выражение

$$A_o = \text{л}_i A., \qquad (3)$$

характеризует набор функциональных признаков, общих для всех объектов данного класса - ядро программного обеспечения.

Общую структуру программного обеспечения, ориентированного на класс объектов и удовлетворяющего двум противоречивым целям - универсальности и эффективности - целесообразно синтезировать на базе принципов, положенных в основу синтеза структуры 0S/360 E116J: функциональной избирательности и функциональной избыточности.

Принцип функциональной избирательности формально отражается выражениями (2) и (3) и означает, что в структуре программного обеспечения конкретной системы управления всегда присутствует ядро A_o, общее для всех объектов данного класса, дополненное некоторым подмножеством $A_к$, характеризующим только данный объект. Таким образом, подмножество функциональных признаков Ад. некоторой микропроцессорной системы управления можно выразить соотношением вида

$$\hat{}_к \sim {}^{A}O\ ^{и}$$

Принцип функциональной избыточности представляет собой основной принцип оптимизации ресурсов при наличии двух или более противоречивых критериев качества. Для микропроцессорного управляющего комплекса такими критериями могут быть

расходы оперативной и постоянной памяти и требуемое быстродействие , надежность и стоимость.

Принцип функциональной избыточности реализуется созданием двух или более программных блоков, работающих по разным принципам, но отражающих один и тот же функциональный признак. Каждый из блоков минимизирован по одному из критериев качества с различной степенью ущерба другим.

Организованное таким образом программное обеспечение представляет собой функционально избирательную и функцио--нально избыточную библиотеку программных модулей (рис.15)

$$B = \bigcup_{i=1}^{I} B_i, \qquad (5)$$

где $B_i = \{b_{i1}, b_{i2}, \ldots, b_{iI}\}$ - подмножество программных модулей, реализующих функциональный признак a_i,

из которой можно генерировать текст ПО для СУ конкретным объектом, что позволяет освободиться от многих повторяющихся этапов разработки программного обеспечения.

3.2. Генерация требуемой конфигурации программного обеспечения из функционально избыточного набора модулей

Исходя из рассмотренного в 3.1. принципа организации программного обеспечения, целесообразно разделить понятия ПО микропроцессорного управляющего комплекса, ориентированного на класс ОУ, и ПО СУ конкретным объектом. Первое из них формально выражается соотношением (5), а второе представляет собой подмножество модулей $B_к \subseteq B$, которое удовлетворяет некоторому критерию» учитывающему в определенной функциональной зависимости частные критерии качества.

Функционально избирательная и функционально избыточная библиотека программных модулей

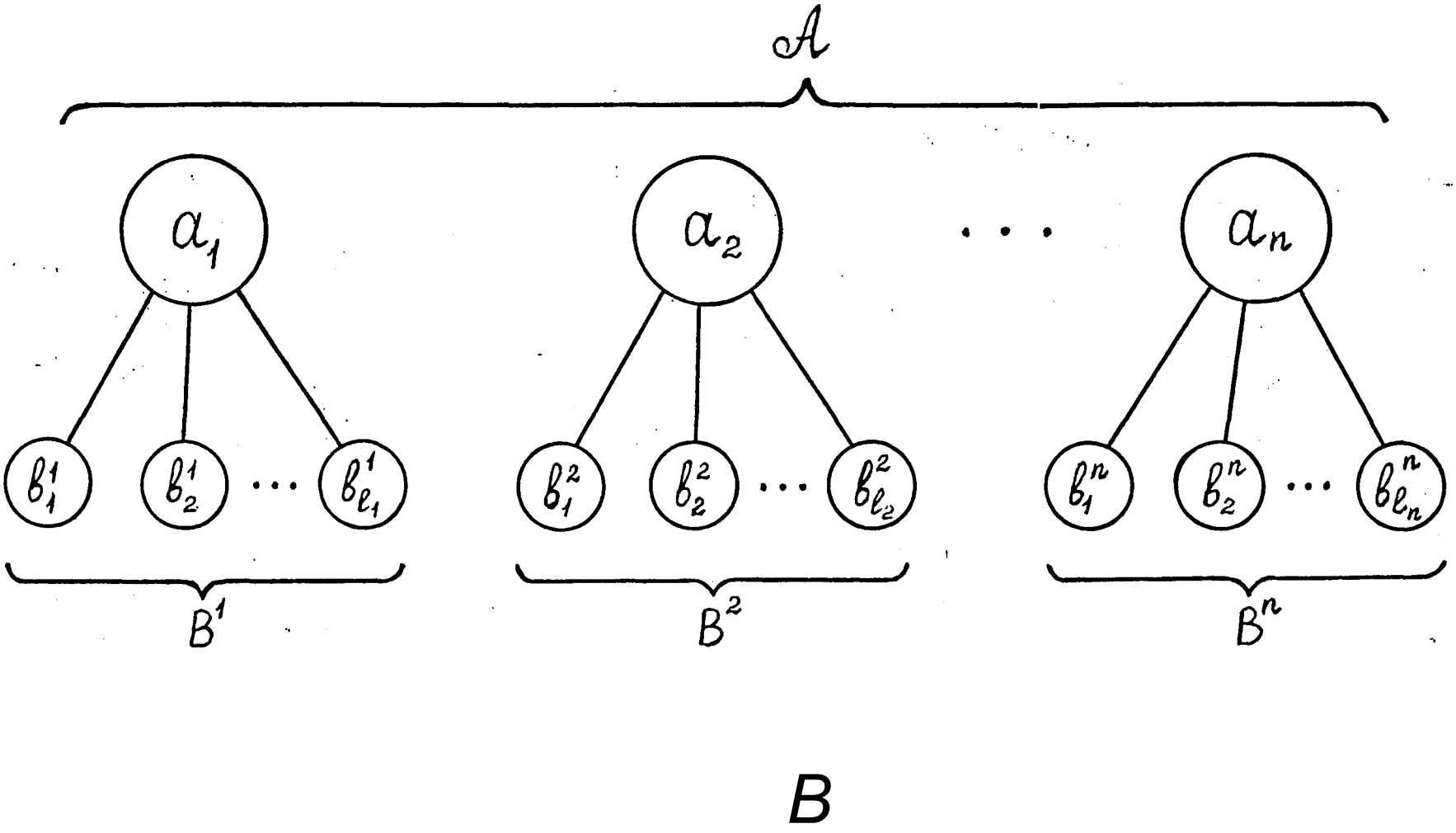


Рис. 15

L

Поиск $B_к$-представляет собой задачу генерации ПО требуемой конфигурации из функционально избыточного набора модулей B.

Одну из возможных моделей генерации можно представить в следующем виде. Пусть:

D = ||cl.J - матрица инциденций функционально избыточного множества B (d. . = 1, если 1-й функциональный
X J
признак имеет j-ю реализацию в виде программного модуля и<1. . = 0- в противном случае),

T ~ " матрица временных характеристик элементов множества B (t. . - время, необходимое для исполнения модуля Ъ*),
и

M = - матрица пространственных характеристик элементов множества B (ц. • - объем памяти,
X J
занимаемый модулем Ъ^),

C = $||_{Gi}.||$ - матрица инциденций множества $B_к$,

где 1 € {1,2,...,к), к - количество функциональных признаков конкретного ОУ, J е {1,2,...,!}, 1 = max 1..
1<I<K

Требуется найти такую матрицу инциденций 0, которая удовлетворяла бы следующим условиям:

$$J_{t_т} = \sum_{i=1}^{K} \sum_{j=1}^{1} W_i i_{13} *_{13}^{mln} \quad (6)$$

при

$$V\ G. .\ A,\ a^{\wedge}, \quad (7)$$
$$j=1 \quad {}_{13} \qquad {}_{1}$$
$$i \qquad к. 1$$

$$1=1 \quad 3=1 \qquad н \quad (8)$$

$C_{\pm}$. & a. ., если (Vr S <1,2.x*J) ($C_{1т}$,= 0);-

^{C}ij 0, если (эс. = 1, i*j) (9>

где J - $_{г^1}$запасы памяти.

. В выражениях (6) и (8) элементы матрицы интерпретируются как десятичные числа, принимающие значения ноля или единицы.

В выражениях (7) и (9) значения С. . интерпретируются как $_{XJ}$ логические переменные.

Аналогично можно построить модель генерации при минимизации занимаемой программным обеспечением памяти, удовлетворяя заранее заданным ограничениям по времени исполнения.

Принцип построения программного обеспечения систем управления, ориентированных на некоторый класс., объектов, формально выраженный в соотношениях (2) - (9), является методологической основой повышения надежности и снижения сроков и соответственно стоимости программного обеспечения.

3.3. Схема генерации и пути решения лингвистических' задач генерации программного обеспечения систем управления гомогенными объектами

Схема генерации программного обеспечения систем управления гомогенными объектами представлена на рис.16.

Процесс проектирования ПО СУ гомогенными объектами должен быть обеспечен мощными и удобными средствами настройки программного обеспечения изделия на конкретные условия эксплуатации, имитационными и испытательными средствами, средствами машинной графики, выпуска технической документации по проекту, эффективными средствами диалога проектировщика и инструментальной системы поддержки процесса проектирования,

воплощенной, в форме АРМ (автоматизированного рабочего места проектировщика).

Абсолютная библиотека программных модулей представляет собой, своего рода, базу данных (БД). Проектировщик всей системы выступает как администратор БД в обычном смысле. Задача генерации рабочей версии программного обеспечения для СУ гомогенными объектами сводится к выбору из ограниченного функционально избыточного множества В подмножества В„ с В, Л

удовлетворяющего некоторому критерию, учитывающему в определенной зависимости характеристики конкретного объекта,

По описанию конкретного объекта осуществляется генерация программного обеспечения для этого объекта (В^) из функционально избыточной и функционально избирательной библиотеки программных модулей В. Причем, описание объекта касается только тех признаков, которые принципиально отличают объекты друг от друга.

Следует отметить, что процесс описания ..и ввода в ЭВМ информации о конкретном объекте управления является наиболее трудоемким. Успешное решение этой задачи конечным пользователем непрограммистом' (инженером-проектировщиком) требует создания специального объектно-ориентированного языка описания объектов (Я00).

Цель создания Я00 - получение эффективного средства для ввода в ЭВМ и корректировки описания, характеристик и параметров объектов. В конечном счете средствами Я00 формируются машинные модели любого объекта автоматизации. Интерпретация, при необходимости, этих объектов средствами машинной графики решает одну из трудоемких задач, состоящую в получении гра-

фической документации.

Основные функции и структура построения Я00 представлены на рис.17, где можно выделить три основные составляющие: формальную грамматику построения языка, программные средства его реализации и дополнительное программное обеспечение преобразования входных данных, описанных на Я00.

Формальная грамматика Я00 описывает его алфавит, синтаксис и семантику и задается в виде грамматики:

$$G = \{ V_{1}, V_{XX}, M, P \},$$

где V_T - терминальный словарь исходных символов языка,

V_H - нетерминальный словарь Я00,

M - начальный символ, определяющий класс всех объектов данной грамматики,.

P - система правил построения грамматики.

При построении грамматики доказывается возможность образования терминального словаря V из допустимых символов.

Семантика Я00 задается набором общепринятых в практике проектирования символов, имеющих конкретное смысловое значение. Множество этих символов задается нетерминальным словарем v_H.

Система правил построения грамматики P задается набором правил вывода любого символа нетерминального словаря V_T в виде цепочки символов терминального словаря V_H. Множество правил вывода P определяет синтаксис грамматики G.

Таким образом, формальная грамматика Я00 позволяет описывать любой объект автоматизации совокупностью предложений в символах словаря V_H V_T и производить семантический и синтаксический контроль правильности вводимой в ЭВМ информации.

Схема генерации программного обеспечения систем управления гомогенными объектами

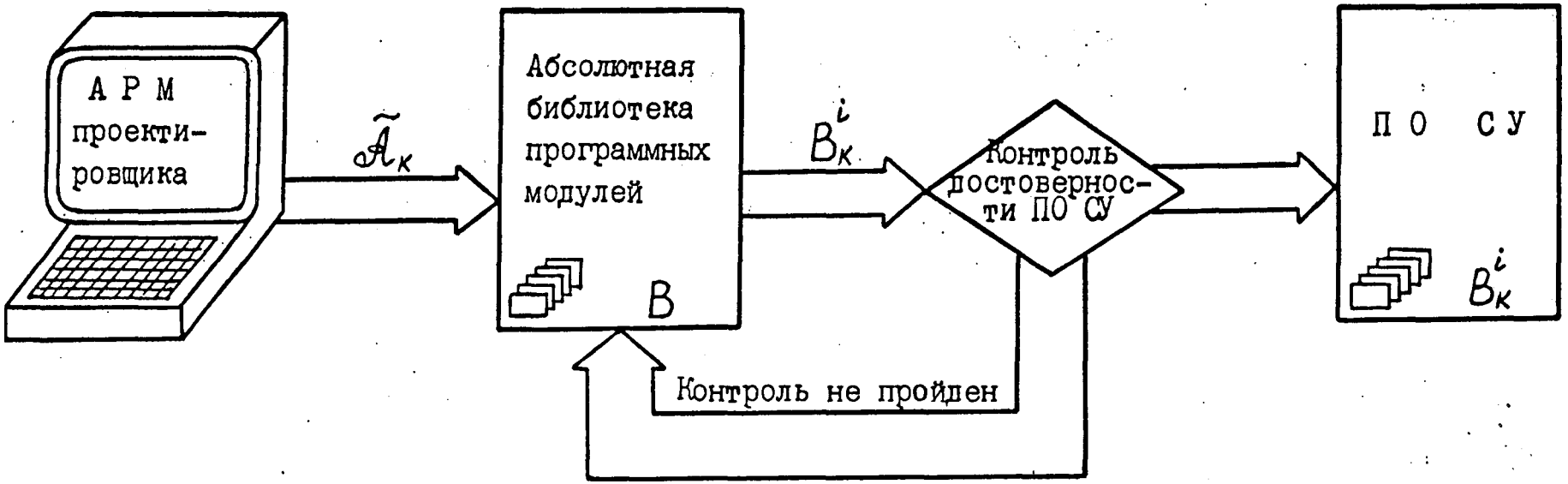
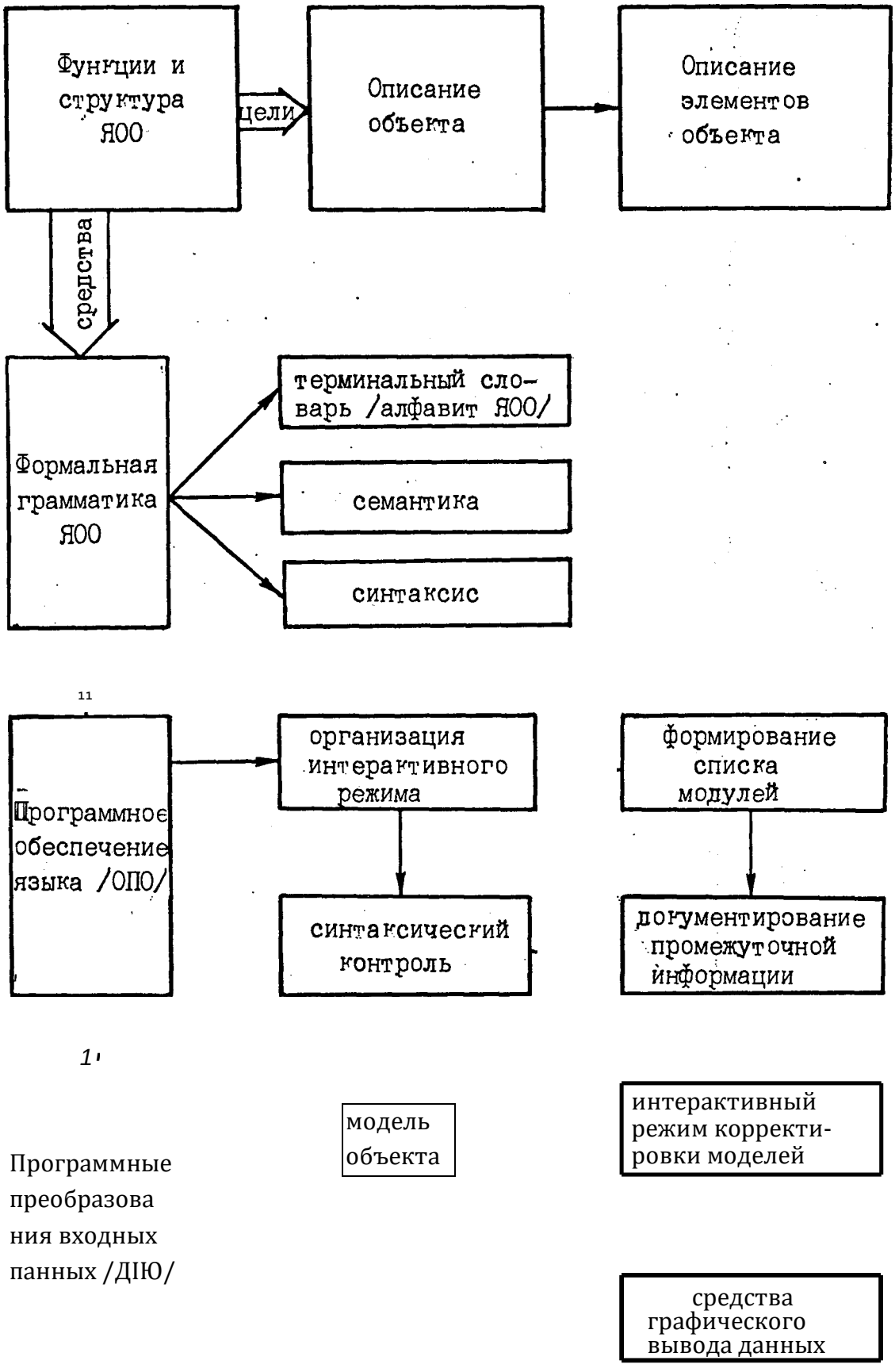


Рис. 18

Основные функции и структура построения ЯОО.



Рис,17

Основное программное обеспечение (ОПО) ЯОО решает задачи, представленные на рис.17. При этом каждому символу словаря V_T ставится в соответствие программный модуль, для которого, при необходимости, дополнительно определяются структурные связи с другими модулями (символами V_T).

В сочетании со средствами организации режима ввода и корректировки информации, синтаксического и семантического контроля ее правильности, а также средствами документирования ОПО решает задачу автоматизации ввода данных.

Дополнительное прогаммное обеспечение (ДПО) ЯОО представляет собой пакет программ для отладки машинной модели объекта, сформированной ОПО. При этом решаются задачи представления различной справочной и графической информации об объекте, интересующей конечного пользователя.

Принципы решения лингвистической поддержки описания объекта нашли свое практическое применение при разработке автоматизированной системы прикладного программирования для систем, управляющих гомогенными объектами.

Единицей измерения передаваемой информации генерации является транзакция. Под транзакцией будем понимать операцию передачи группы данных по одному обращению [43]. Причем, согласно принятым в 2.2.2. определениям ПО, данными являются и программы, и метаданные и собственно данные. То есть длина передаваемых данных может быть различной. Единственным источником ошибок (абстрагируясь от аппаратных сбоев) является искажение информации при транзакции из абсолютной библиотеки в файл ПО СУ. Таким образом, при осуществлении транзакции из функционально избыточной и функционально избирательной биб-

лиотеки программных модулей в библиотеку программного обеспечения конкретной системы управления возникает проблема достоверности передаваемых данных.

Достоверность информации позволяет оценить соответствие принятого сообщения переданному [99]. При непройденном "п"-кратном контроле достоверности группы передаваемых данных возникает необходимость повторной собственно транзакции из абсолютной библиотеки программных модулей. Контроль достоверности ПО может осуществляться различными способами.

3.4. Контроль достоверности программного обеспечения

Физической средой, являющейся носителем информации о внутреннем состоянии управляющей системы, является запоминающее устройство (ЗУ). Следовательно, реализация предложенного принципа контроля достоверности ПО должна предусматривать контроль идентичности хранящейся.в ЗУ информации. Нарушение идентичности содержимого памяти будет свидетельствовать о недостоверности ПО.

Для контроля достоверности передаваемой информации необходимо применять методы, позволяющие сократить объем диагностируемой информации, т.е. производить обмен со сравнением не содержимым памяти, а контрольными словами, размер которых значительно меньше, чем размер .страницы памяти. Для этого каждый разработанный в опытном образце модуль необходимо снабдить контрольными суммами.

Пусть имеется страница памяти, содержащая N двоичных n-разрядных слов. Совокупность значений разрядов 1-го слова

определит состояние слова, которое обозначим Упорядоченная последовательность (Ц,!^,!^,... ,h_N) определит состояние страницы памяти. Перенумеруем все возможные последовательности слов, определяющие различные состояния памяти и обозначим 1-ую последовательность через {11} $_i$. Условие достоверности ПО определится {h} .= {11} . (1 * J). Недостоверное сос-
J- J
тояние ПО будет характеризоваться преобразованием {Ы^—> {П} . (1 * 3), где! {11}. - состояние страницы памяти, соответствующее достоверному ПО; {11}. - действительное текущее
V
состояние памяти.

Метод контрольного суммирования основан на преобразовании N информационных слов в N* контрольных двоичных слов. Данное преобразование характеризуется контрольной функцией К. Область определения контрольной функции - множество всех возможных состояний страницы памяти, а область значений - множество состояний контрольного слова. Таким образом, не достоверной информация будет при условии

К ({Юр = {г}$_{\pm}$, "

К ({Ю .) = {г}.,
d d

если {г} i *

Рассмотрим контрольную функцию вида

К ({Ы$_{\pm}$) = S h. (mod ц), (10)

которая представляет собой известный метод контрольного суммирования с циклическим, переносом [65].. -

Будем считать, что суммирование производится по модулю $2^{П}$, то есть число разрядов контрольного слова равно числу разрядов слова памяти;'

$t_{Q(5}$ - время обмена со сравнением одного п - разрядного

слова;

N $t_0$$ - время обмена со сравнением одной страницы длиной N слов;

$t_{„р}$ - время работы программы суммирования при сложении одного слова;

$(N—1)t_{КС}$ - время работы программы суммирования при сложении N слов памяти.

Применение метода контрольного суммирования позволяет получить значительный выигрыш по времени использования программы диагностики состояний ЗУ, причем данный метод особенно эффективен при больших объемах контролируемой памяти.

Снижая потребность в избыточных временных ресурсах, необходимых для выполнения алгоритма диагностики, метод контрольного суммирования приводит к уменьшению достоверности контроля. При арифметическом суммировании считываемые из ЗУ данные суммируются по некоторому модулю р, причем размер страницы памяти может значительно превышать длину контрольного слова. Вследствие этого одной контрольной сумме будет соответствовать множество информационных последовательностей данных. Таким образом, появляется вероятность необнаружения ошибки g , т.е. значение контрольного слова недостоверного ПО может оказаться равным значениям контрольных слов достоверного ПО.

В качестве показателя достоверности контроля примем вероятность обнаружения ошибок В. Для оценки значения примем следующие допущения:

ошибки различных 'типов равновероятны, т.е. переход от состояния страницы памяти Ш. к {Ы • происходит с одинако-

вой вероятностью;

сбои в контрольных словах не возникают, так.как количество контрольных слов много меньше информационных.

Обозначим через H_N - число всех возможных состояний страницы памяти, состоящей из N n-разрядных слов, а через H_N - число состояний памяти, сумма которых по модулю равняется, S, 0 < S < р.—1. Тогда вероятность необнаружения ошибки определится выражением

$$\$ОШ = W^{S)\cdot 1/(H}N'' \ ^{1Л}$$

а величина Р :

$$P = 1 \text{ -----} \frac{jp-r-r-}{} \quad (И)$$

Очевидно, что число всех' возможных состояний страницы памяти из N двоичных n-разрядных слов будет равна $H_N = 2^{\wedge}$.

Исследуем величину $H_м$ ..(S). Известно [921, что при числовом контроле по модулю код заданного числа S представляет собой наименьший положительный остаток от деления числа на выбранный модуль:

$$S = S - Kx|i \quad (12)$$

где К = [S/ц], S є [0,ц-1]. В нашем случае S - это сумма всех информационных слов страницы памяти: .

$$S = \sum_{i=1}^{N} h_i, \ h. - {}_1 \text{целое}, \ Q < h. \ I \ {}_1 2^{П} \ 1 \quad (13$$

Максимальное значение суммы N n-разрядных чисел равно $N(2^n-1)$, тогда значение к в выражении (12) может принимать значения к = 0,1.2,...,к* = /(N(2^n-1)-S)/р,.

Решение уравнения (13) в целых числах относительно h_i будет представлять собой функцию $1_N(S)$, которая для каждого значения суммы S определяет число состояний страницы памяти из N чисел, сумма которых равна S. С учетом (12) имеем:

$$^{H}N,p.(\wedge = {}^{1}N(\wedge) + V\$ + I^{i}) + V\$ + {}^{2}H) + ---+VS * K*Ц) =$$

$$= \sum_{K=0}^{14} 1_N(S + K|X) \qquad (14)$$

Покажем, каким образом рассчитываются значения функции $1_N(S)$. Очевидно, что $l_N(0) = 1$, так как сумма N неотрицательных чисел будет равна нулю только при нулевом значении всех слагаемых. Рассмотрим страницу памяти, имеющей размер в одно n-разрядное слово - N =1. При этом значение суммы окажется равным самому значению слова hp а $1_1(S)$ определится 'выражением

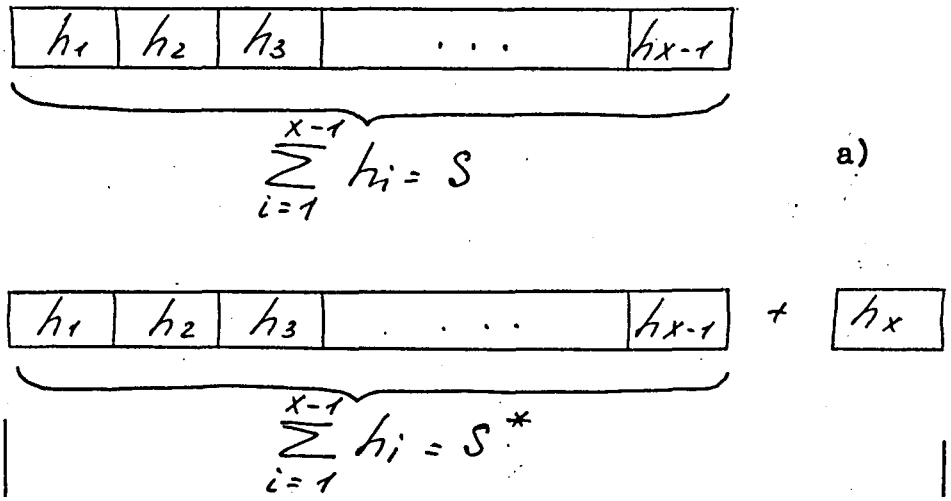
$$1\ (S) = \begin{cases} 1, & \text{если } 0 \le S < 2^{П} - 1 \\ 0, & \text{если } S < 0 \\ 0, & \text{если } S > 2^{П} - 1 \end{cases} \qquad (15)$$

Рассмотрим информационную последовательность из (x - 1) слов (рис. 18). Значение суммы ее элементов $S = \sum_{i=1}^{x-1} J1_L$. Добавим к массиву еще одно слово h^, значение которого может быть от О до 2^n -1 (рис.18). Чтобы массив из х слов имел такую же сумму S, как и массив из (х-1) слов, необходимо выполнение следующего условия: сумма первых (х-1) слов должна иметь такое значение S*, что $S-2^n+1 < S < S$, тогда $S^*+h_x = S$. На основании этих рассуждений следует вывод выражения для расчета $1_N(S)$, который представлен в таблице 1. Таким образом получено выражение для определения числа состояний памяти, имеющих одинаковую сумму:

$$i_N(S) = VJS) + [1_{x\_г}(5-1) + WS-2) + \ldots + \qquad (1б)$$

$$+ l_{x\_j}\ (S-2^n+1)\ ]$$

Состояние страницы памяти



Рис,18

Таблица 1

Определение числа состояний страницы памяти ..

Значение /ix	Сумма массива из C*"*) слов	Число информационных последовательностей, имеющих сумму
0	S	
S	S-/	
2.	S.-2	~-f (S> -2)
i		
t \	1 • *	ъ * *
2^n-i		4-/4-^ г,)

4 (S)-- 4-г(.$)*

Учитывая, что значений суммы, выделенной квадратными скобками, равно 1 (5-I) - 1 (S-2^n) представим выражение в виде

$$1\_.(S) = 1\_\_(S) + \quad - 1\ ,.(S-2^n) \qquad (17)$$

-А. -Л» ± Д «А. X

Таким образом, для вычисления точного значения вероятности достоверного обнаружения ошибок необходимо:

а) по формуле (17) рассчитать значения $1_N(S)$ для S = 0,1,2 N(2^n-1);

б) используя (17), определить значения H_N ^(S);

в) подставив полученные значения H_N ^(S) в (И), определить Р.

В работе [99] приводится приближенная формула для расчета $1_N(S)$ при больших N. При этом состояние•1-го слова рассматривается как случайная величина, равномерно распределенная на интервале [0,2^n- П. Согласно центральной предельной теоремы, величина £ h_i= S для больших N распределена по нормальному закону. Тогда значение ^(S)^1311 есть вероятность того, что сумма N -элементной последовательности данных равна S, или

$$lUS)/^{N} = -^{-------}\ {}_e-(S-a)^2/2S^2 \qquad (18)$$

6 2П

где: ***a*** = [N(2^n-l)/2]; 6 * N(2^n-10).

Выражение (18) позволяет просто рассчитать значения $1_N(S)$ функции и при больших N обеспечивает достаточно высокую точность значения Р [65].

Исследуем основные свойства функции P(n,N,|i,S). На рис.19 приведены кривые, характеризующие ее поведение при

Вероятность достоверного контроля суммированием
при фиксированных /2 и $/У$.

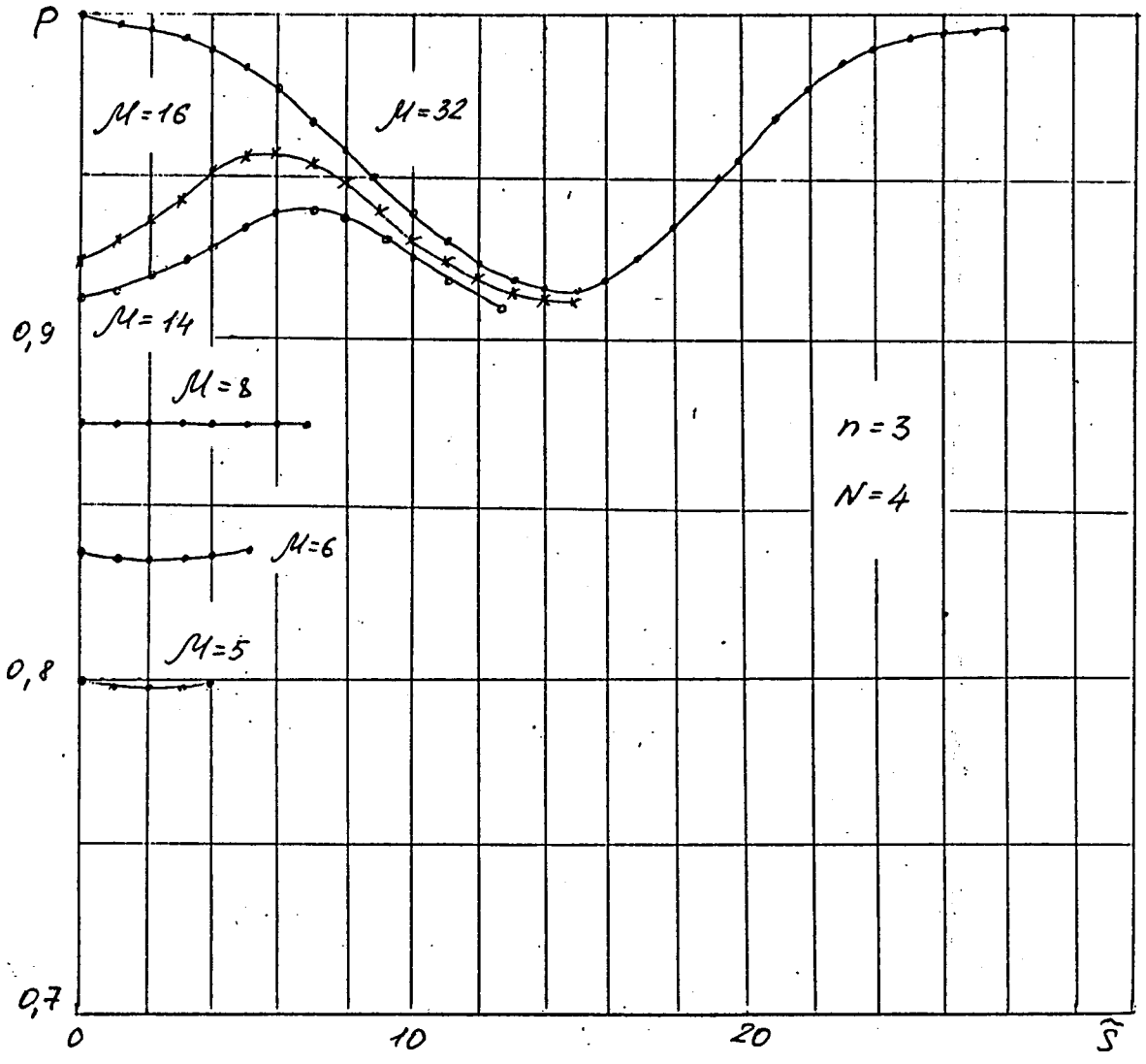


Рис.19

'Заметим, что при модуле суммирования $p_1=2^n$ (на рис. 19 - p. =8), то есть при равном числе разрядов контрольного слова и слова памяти, функция P(n,N,p,,S). не зависит от значения контрольного слова S. Докажем это положение.

Рассматривая совместно (14) и (17), запишем

$$\sum_{k=0}^{k^*} 1_{(S + K|1)} = \sum_{k=0}^{k^*} (S_1 + кр.) + \sum_{k=0}^{k^* \text{ тм}} ((S - 1)\text{mod}p. + кр.) - \sum_{k=0}^{1} 1_T <((S \text{ “ } 2^n)\text{mod}p. + кр.),$$

или

$$H_N\hat{}(S) = H_{N\_1>P1}(S) + H_N\hat{}((S\text{-}1)\text{mod}p - H_{N>(u1}((S\text{-}2^n)\text{mod}p. \quad (19)$$

Из того, что $(S - 2^n)\text{mod } 2^n = S$ следует:

$$H_{N\text{-}l,2^n\hat{}}(S) = H_{N\text{-}1,2^1\hat{}}((S\text{-}1)\text{mod } 2^n.$$

Значит $H_{N,2^n}(S.) = H_{N,2^n}„((S\text{-}1)\text{mod } 2^n$, то есть ..число состояний страницы памяти, сумма элементов которой по модулю p. = $2^П$ равняется S, будет одинаковым при любом S, где S = 0,1, 2...,$2^П$ - 1. Значение P при p. =$2^П$ определим следующим образом

$$p = 1 - \frac{_9nN}{2^П * 2^{тм}} = 1 - {}^{n\ nN}{}_2 \quad (20)$$

Исследуем характер изменения функции P(n,N,p.,S) при фиксированных п,р, (рис.20). Анализируя поведение кривых, приходим к выводу, что с увеличением N их амплитуда уменьшается и при p. « $N(2^n\text{-}1)$ стремится к значению P = 1-1/p.. Действительно, при p. « $N(2^n\text{-}1)$ для получения′ значений $H._т$. (S) суммируется большое число величин 1 (S + кр,), распо-

14 9 fwl И

ложенных во всем диапазоне изменения S с шагом p_1. Поэтому $H_{и>ц}(5)$ = % /Ц и P = 1-1/ц.

Вероятность достоверного контроля суммированием при фиксированных n и N.

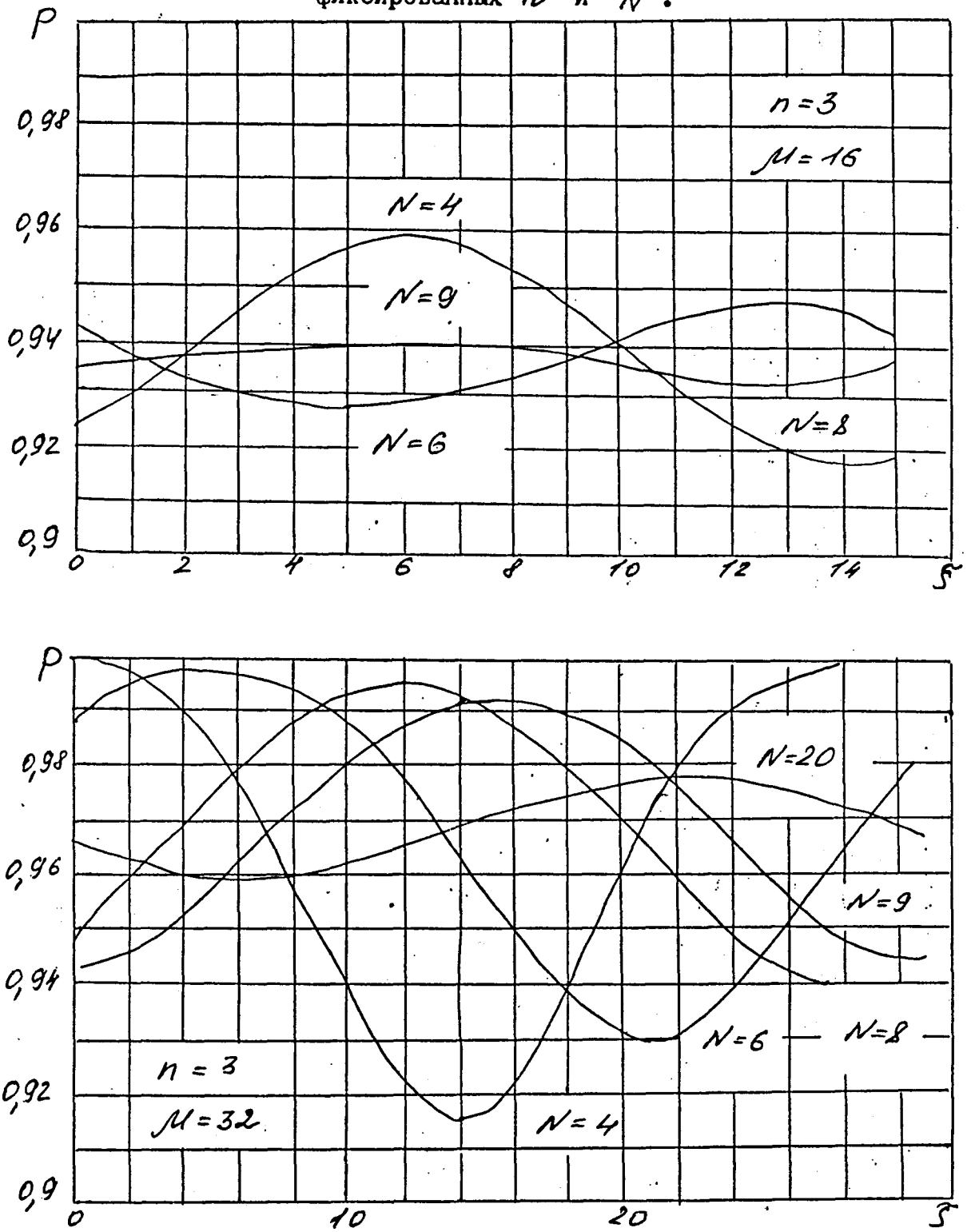


Рис. 20

Реализация контрольной функции вида (10) производится программными средствами. При этом для упрощения программы подсчета контрольных слов целесообразно выбирать модуль суммирования р, = 2^, где п - разрядность хранимых в ЗУ данных, п - число контрольных слов в контрольной сумме.

В таблице 2 приведены значения вероятностей обнаружения ошибок Р в зависимости от разрядности слов при суммировании по модулю р, = $2^{П}$ [65].

Таблица 2

Вероятности обнаружения ошибок

n	2	8	16	32
p	0,9375	0,996094	0,999985	"I,000

Вероятность обнаружения ошибок при суммировании по модулю $2^{П}$ высока, если разрядность слова п > 16. Однако разрядность широко распространенных микропроцессоров п = 8. Суммирование по модулю р, = 2^{8} = 256 не обеспечивает достаточной вероятности: р = 0,996094. Для повышения достоверности в подобных случаях необходимо использовать модуль суммирования р. = $2^{2п}$. Вероятность обнаружения ошибки в этом случае равна Р = 0,999985, что является вполне приемлемым.

Таким образом, для контроля.достоверности передаваемой в ПЗУ информации целесообразно использовать метод контрольного суммирования. Для этого каждый разработанный программный модуль в опытном образце необходимо снабдить контрольной суммой. Причем, независимо от выбранного типа ПЗУ (масочные, однократно - программируемые, репрограммируемые), в техноло-

•nwtTQr"rrryй' TTC'TTfflUvo тг/тиФТУл ттт. ттппп'тпэа'п'иплфтх ТТЛ /-чЛагэпФо потт

Л XX 1. V k/IVJXL .ЦиХХМ 11VU XU\^XXXJ^/^»/XX^ X

3.5. ВЫВОДЫ К ГЛАВЕ 3

1. В общей технологии создания микропроцессорной системы управления проектирование конкретной конфигурации программного обеспечения становится основным по трудоемкости и длительности этапом. Сокращение сроков разработки как специального программного обеспечения, осуществляющего основные целевые функции управления, так и средств автоматизации программирования, становится реальным, если программное обеспечение ориентировано не на конкретный объект, а на некоторый класс объектов, выделенных в класс гомогенных.

Разработаны основные принципы генерации программного обеспечения для систем управления гомогенными объектами.

1. Единую для выделенных объектов технологию проектирования и общую структуру программного обеспечения, ориетированные на класс объектов, предложено синтезировать на базе принципов функциональной избирательности и фукциональной избыточности, контроле достоверности программного обеспечения.

3. Разработана схема генерации программного обеспечения систем управления гомогенными объектами, в которой по описанию конкретного объекта осуществляется собственно генерация ПО для объекта из функционально избирательной и функционально избыточной абсолютной библиотеки программных модулей.

4. Разработаны пути , решения лингвистических задач генерации программного обеспечения систем управления гомогенными объектами. Лингвистические задачи генерации программного обеспечения сведены к решению вопроса о выборе правил описания объекта и средств его описания.

5. Предложены методы контроля достоверности тиражируемого программного обеспечения, основанные на контрольном суммировании.

4.АВТОМАТИЗАЦИЯ ПРИКЛАДНОГО ПРОГРАММИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ ГОМОГЕННЫМИ ОБЪЕКТАМИ

4.1. Современное состояние МПЦ

Предложенные технологические принципы проектирования программного обеспечения для систем управления гомогенными объектами могут быть использованы при разработке микропроцессорной системы управления стрелками и сигналами (микропроцессорной централизации).

Внедрение средств вычислительной техники в системы железнодорожной автоматики в настоящее время проходит первый этап - этап эксплуатации экспериментальных образцов и исследование возможностей более широкого использования подобных систем. Это связано с тем, что средства с необходимыми надежностными и технико-экономическими параметрами стали возможны только благодаря появлению микропроцессоров (первые мировые образцы появились в 1971 г.) и ЭВМ на их базе.

В СНГ в настоящее время отсутствуют не только промышленно освоенные, но и экспериментальные образцы систем компьютерной централизации, сигнализации и блокировки.

Имеются сведения о вводе в эксплуатацию ряда подобных систем в зарубежных странах [1,32,53,56,1013. Так, в 1985 году была пущена на станции Лемингтон Спа в Великобритании система централизации с применением твердотельных элементов фирмы Дженерал Сигнал Лимитед. Известна отказоустойчивая вычислительная система с тремя симметричными вычислительными машинами, использующаяся в системах диспетчерской централи-

зации на железных дорогах Японии. Подобные работы ведутся в ФРГ (SIMIS-C, ESTW L90), а также в США, Дании, Франции. В СНГ ведущей организацией, работающей в области создания компьютерной системы централизации, является Гипротранссигналсвязь (г.Санкт-Петербург).

Подходы к созданию таких систем в общем различны, однако главным показателем является использование избыточности для обеспечения высокого уровня надежности и безопасности, что является непременным условием для внедрения подобной техники на железнодорожном транспорте.

Микропроцессорная централизация (МПЦ) представляет собой сложную управляющую систему, построенную на базе вычислительной техники. Она отличается от известных систем электрической централизации принципиально новой элементной базой, значительно большими функциональными возможностями, большим числом компонентов и более сложными связями между ними.

МПЦ является системой централизованного управления движением поездов на железнодорожных станциях, обеспечивающих повышение пропускной и провозной способности за счет уменьшения времени установки маршрута и безопасности движения - исключением недопустимых состояний системы при наличии отказов и сбоев оборудования и относится к классу высоконадежных отказоустойчивых систем управления, сбои и отказы оборудования в которых могут привести к человеческим жертвам или нанести огромный ущерб.

МПЦ позволит обеспечить:

маршрутное управление с накоплением маршрутов;

выбор наиболее оптимальной трассы маршрута с созданием

наименьшей враждебности другим маршрутам;

документирование данных о поездном положении;

выдачу информации о выполнении и нарушении графика движения;

автоматизацию по заданной программе маневровых передвижений;

выдачу информации оперативным работникам станции;

получение дополнительных сведений и данных от других подсистем,а также снабжение их необходимыми данными о поездном положении на станции.

Анализ результатов исследований, выполненных ХарГАВДТ, [16,44,120], а также некоторых зарубежных децентрализованных мультимикропроцессорных систем [1,32,53,56,101,109,1291 позволяет к источникам экономической эффективности МПЦ отнести следующие:

расширение функциональных, возможностей системы централизации;

возможность применения современных средств отображения;

идентичность аппаратного и программного обеспечения для связей с системами АСУЖТ верхнего уровня;

относительная простота изменения систем управления и отображения при изменении путевого развития станции;

значительная экономия производственных площадей;

значительная экономия активных материалов и электроэнергии;

стандартизация элементов железнодорожной автоматики и телемеханики путем создания унифицированных программных и аппаратных модулей;

возможность автоматизации унифицированных модулей системы управления стрелками и сигналами;

реализация алгоритмов практически любой сложности с меньшими затратами;

автоматизация и оптимизация управления работой станции (выполнение информационно-планирующих функций, связанных с обработкой ***и*** отображением оперативно-технологических данных по приему и отправлению поездов, а также функции автоматизации работы технической конторы станции);

упрощение обслуживания.

4.2. Технология проектирования МПЦ и надежность

Достоинства микропроцессорной централизации представляют собой условия хотя и необходимые для обеспечения ее жизнеспособности, но не достаточные. Действительно, низкие показатели ряда эксплуатационных характеристик МПЦ и, в первую очередь, ее надежности и безопасности, могут свести к нулю эффект от ее использования.

Как следует из главы 1, проблема обеспечения требуемого уровня надежности в управляющих вычислительных системах, вообще, и в МПЦ, в частности,, является наиболее сложной. Сложность проблемы обусловлена появлением ряда специфических особенностей, отличающих ее от проблемы традиционных релейных систем управления стрелками и сигналами. Действительно, если в электрической централизации безопасность и надежность достигается путем разработки и тщательной проверки механических конструкций и за счет особого внимания к обеспечению заданных электрических характеристик, то при проектировании

устройств сигнализации МПЦ задачей разработчиков становится тщательный подбор электронных компонентов системы и создание высоконадежных алгоритмов и программ. Надежность системы МПЦ в целом будет определяться:

надежностью аппаратных средств;

надежностью программного обеспечения;

безошибочностью работы оператора;

надежностью самого объекта управления.

Таким образом, если в традиционных системах проблема надежности замыкается только в пределах надежности аппаратуры, то в МПЦ, кроме надежности аппаратуры, огромную, если не решающую, роль играет надежность программного обеспечения.

Ошибки в программном обеспечении МПЦ могут вести к катастрофическим последствиям. Это обстоятельство определяет максимальное использование всех методологических и технологических ресурсов в интересах повышения надежности. В этом случае общую стратегию проектирования программного обеспечения МПЦ можно представить в еле,дующем виде: задается минимально допустимый уровень надежности и тестирование и отладка ведутся до тех пор, пока этот уровень не будет достигнут. Очевидно,что стоимость ПО как изделия будет зависеть от технологии его , проектирования .

4.3. Железнодорожная станция в МПЦ как объект управления, относящийся к классу гомогенных

Особо важное значение при построении высоконадежных управляющих систем (к каковым относится микропроцессорная централизация) приобретает'разработка технологии проектирования

программного обеспечения. При выборе необходимой ТП из всего множества возможных следует различать понятия

ТП как описание процесса программирования (своего рода ''know how"),

ТП как сам процесс создания программного обеспечения во времени (глава 1).

ТП как макротехнология разработки всех программных продуктов (''know how") для всех станций будет единой. ТП же как сам процесс создания ПО во времени (с учетом методологии) зависит от

характера объектов, для которых разрабатывается программный продукт,

от ориентации создаваемого программного обеспечения (на конкретный объект управления или некоторый класс объектов).

Для станций, как некоторого класса объектов управления, задача разработки программного обеспечения может быть..сформулирована как задача создания единой ТП, у которой - минимум повторяемых технологических этапов во времени. Докажем это.

Рассмотрим станцию как объект управления микропроцессорной централизации и определим к какому типу объектов она относится:

уникальным,

равным,

гомогенным.

Допустим, что станции являются равными объектами. Тогда и разрабатываемая для них технология проектирования МПЦ должна быть единой.

Однако, на практике не существует двух абсолютно одинаковых станций и станции строго можно было бы отнести к уникальным (различным) объектам. Тогда для каждой станции необходимым представляется создание уникальной объектно- ориентированной микропроцессорной централизации. По этому пути действовали первые отечественные разработчики микропроцессорной централизации. Первоочередной виделась задача создания действующего макета МПЦ, ориентированной на конкретную станцию .

Однако, нетрудно заметить, что станции сочетают в себе черты как уникальных, так и одинаковых объектов. Они характеризуются некой функциональной однородностью. Они представляют собой объекты,, управляемые по одним и тем же законам, но отличающиеся друг от друга количеством стрелок, светофоров, рельсовых цепей. Эти объекты названы гомогенными (глава 2).

Пусть В = (to ,Ъ$_0$,...,Ъ„} - множество станций - объектов,

1 ***С.*** дл

• обладающих функциональной однородностью, с

{c_1,c_2,...,c_m,F) - допустимое множество типов элементов (рельсовых цепей, стрелок, светофоров), принципиально присущих любой из станций « В, 1=ТД, F={f_1,f_2,...}; $Г_i$ = {c^^,c_2,... ,c_m) - допустимый набор топологий станций, Dj = ... Л??[1],- топология конкретной, станции (множество признаков станции Ъ±), а Р = {p_1, p_2,..., р^) - множество функционально законченных, текстов ПО, удовлетворяющих условию р.~ Ь.. Тогда, если <->D.=QD., 1 =17ТГ, то р.=р., 11 1 1 ***3*** 3 1 3

что свидетельствует о достаточности единой технологии проектирования и разработки ПО для любой из станций Ь.

из станций b_i находятся в соотношении (УieТГД)) (з>ТГД)) (Р-.хр) И В предельном случае (vi, j « (fX), 1*3) (р.*р.)
X V J. J
являются уникальными.

Воспользуемся математической моделью класса гомогенных объектов, приведенной в 2.1, и докажем, что ж.д. станция относится к выделенному классу и к ней, как объекту управления МПЦ может, быть применена выработанная для гомогенных объектов стратегия проектирования ПО.

Железнодорожную станцию определим как объект, содержащий упорядоченные совокупности элементов, а именно: рельсовых цепей (р.ц.), стрелок, светофоров, соединенных между собой по определенным правилам, и управляемых по одним общим законам.

Другими словами, железнодорожные станции

содержат единый набор элементов: рельсовых цепей, стрелок, светофоров,

имеют одну область применения(назначение и цели одинаковы),

имеют идентичную архитектуру, то есть принципы построения связей между элементами едины и определены,

управляются по одним общим законам, то, есть функции управления одинаковы.

Однако, имеются и различия. Для двух- станций могут быть различными:

размер станции, то есть количество содержащихся элементов (рельсовых цепей,светофоров, стрелок),

типы имеющихся элементов,

функциональные связи между элементами , а соответственно

и - возможные маршруты, таблицы враждебности и тому подобное.

Исходя из выше изложенного, можно сделать вывод о принадлежности железнодорожной станции к классу гомогенных объектов и, соответственно, о:

применении к МПЦ предлагаемой стратегии проектирования программного обеспечения;

реализациии таких технологических процессов, в которых большая часть технологических этапов - общая для всех станций.

4.4. Оценка стоимости разработки технологии проектирования МПЦ

Для доказательства экономической целесообразности разработки технологии проектирования МПЦ, ориентированной на множество объектов, необходимым представляется произвести оценку затрат на программирование МПЦ.

1.Оценка стоимости разработки технологии проектирования, ориентированной на конкретный объект.

Пусть имеется 1 объект управления - станция. Тогда

$$s = \sum_{i=1} s_i,$$

где: S - общая стоимость разработки ПО;

- стоимость этапов разработки технологии программирования, соотве тственно:

S_1 - стоимость этапа внешнего проектирования,

S_2 - стоимость этапа проектирования,

- стоимость этапа кодирования,

- стоимость этапа тестирования и отладки

S$ - стоимость этапа сопровождения.

Причем, при разработке ПО МПЦ в целях повышения уровня надежности можно говорить о повышении роли внешних спецификаций, проектирования, кодирования, отладки и тестирования и о значительном сокращении этапа сопровождения. Это ведет к снижению стоимости устранения ошибок, возникающих при разработке ПО МПЦ. по сравнению с системами обработки данных (рис.8), а, следовательно, к снижению общих затрат на создание ПО МПЦ.

2. Оценка затрат на разработку технологии программирования для "п" станций.

2.1. Оценка стоимости разработки технологии программирования для "п" объектов управления, исходя из создания "п" технологий программирования.

Пусть имеется "п" станций.

S^-Sj - стоимость этапов разработки для 1-го объекта управления,

S - общая стоимость разработки.

Будем исходить из того, что станции представляют собой уникальные объекты управления. Тогда для создания технологии программирования для "п" объектов управления, исходя из разработки для каждого из п объектов новой технологии:

$$s = si + s' + si + s; + s^{*}+...+ sj + si + s^{*} +$$

$$+ sj + sj +...+s? +s? + S? + S^{*}J + sg =$$

$$= E\,(\,s_1 + s_2 + s, + s, + s, \,) = E\,E\,\bar{sJ}$$

Пусть имеется ″п" объектов управления - станций:

- стоимость этапов разработки для 1-го объекта управления,

S* - стоимость уникальной части ПО.

Предполагается, что для каждого из ″п″ объектов управления

$$S' = Si + S',$$

где: - стоимость тестирования уникальной части,

S£ - стоимость сопровождения уникальной части,

S - общая стоимость разработки ПО,

S_Q - стоимость системного ПО для единой универсальной технологии программирования,

S_n - стоимость прикладного ПО для п станций.

При оценке стоимости разработки единой для всех п станций технологии программирования необходимо учитывать введе-.ныэъв разделе 2 понятия системного и прикладного ПО, понятия программы вообще. Тогда

$$S=S_{c+}S_n$$

$$^{s}c=.2^{s}i$$

$$1=1$$

$$S_n = \blacksquare\ (\&i + S')n$$

Отсюда $S = S_1 + S_2 + S^{\wedge} + S_4 + S_5 + (Si + S£)n$,

то есть стоимость уникальной части для одной станции соста-

что соизмеримо с общей стоимостью разработки единой технологии программирования.

Что касается стоимости уникальной части, то с ростом ″п′′

S′ и S_n снижается (рис.21), так как можно говорить о переносе разработанной методики тестирования на другие объекты, о накоплении опыта тестирования и отладки, что позволит свести количество ошибок в информационной части к mln (величина же S$ весьма незначительна).

Представленная оценка стоимости разработки уникальных технологий программирования для станций (2.1) и единой для станций технологии (2.2) дает возможность сделать вывод об экономической целесообразности разработки единой технологии программирования для всего множества станций.

Стоимость разработки ПО можно рассчитать также, исходя из:

норм выработки программиста, его средней зарплаты и общего объема памяти, занимаемого ПО;

стоимости одного байта системного и прикладного ПО и общего объема памяти, занимаемого ПО.

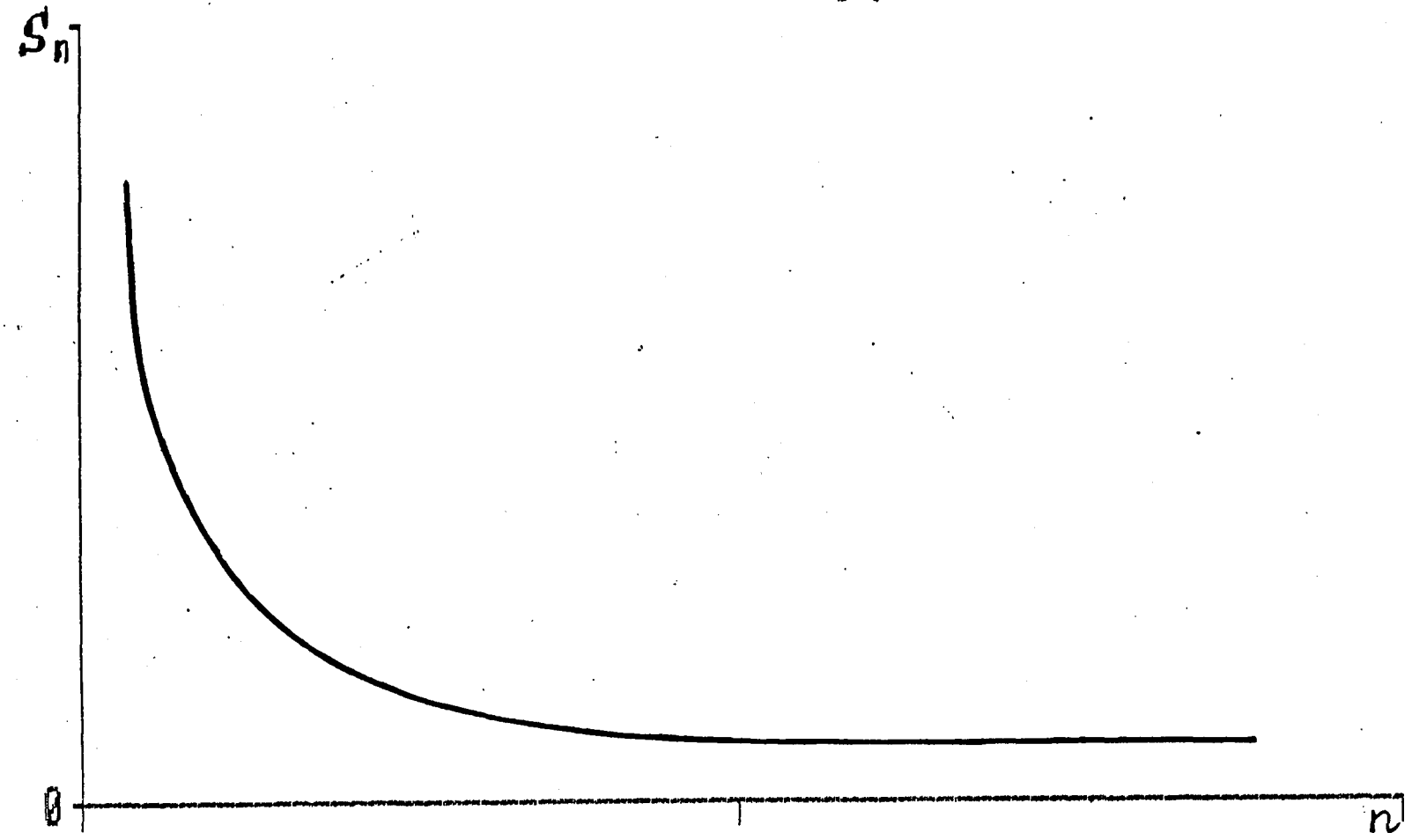
4.5. Технология проектирования программного обеспечения микропроцессорной централизации

Микропроцессорная централизация обладает рядом особенностей, стимулирующих поиск новых технологических принципов. К ним следует отнести:

принадлежность железнодорожной станции к классу гомогенных объектов;

принадлежность МПЦ к высоконадежным управляющим системам.

Зависимость стоимости разработки прикладного ПО от количества объектов управления



Рис, 21

Технологические же 'принципы проектирования программного обеспечения МПЦ, использовавшиеся на прошедшем этапе, могут быть оправданы только новизной проблемы. Разработка велась для конкретной станции и носила "поисковый", экспериментальный характер, без учета использования системы в будущем на какой-либо другой станции. Такие аспекты технологии проектирования, как повышение надежности и снижение стоимости разрабатываемого ПО МПЦ не затрагивались вообще, что абсолютно бесперспективно.

При использовании для МПЦ выработанной стратегии проектирования программного обеспечения гомогенными объектами применяется терминология, предложенная в главе 2, а именно: программа - совокупность команд (логическая часть программы), интерпретируемых процессором управляющего вычислительного комплекса, и сопровождающие их константы (информационная часть программы); программное обеспечение системы управления; внутреннее ПО (тексты на внутреннем языке ЭВМ); ПО объекта управления (часть внутреннего ПО, поставляемого с ЭВМ на конкретный объект управления); системное ПО СУ (часть внутреннего ПО, общая для всех объектов данного класса); прикладное ПО объекта управления (уникальная составляющая ПО ЭВМ). Графически программа представлена на рис. 14 (глава 2).

Кроме того, всю информацию системы МПЦ по степени ее постоянности (неизменности) можно разделить на следующие группы:

постоянная информация (нормативно-справочная);

условно-постоянная;.

переменная.

Постоянная информация - данные общие для любой станции, не зависящие ни от конкретной станции, ни от процессов, протекающих на ней во время управления. Таковыми являются, например, коды директив, их числовые эквиваленты, тексты сообщений.

Условно-постоянная информация зависит от конкретной станции, но не зависит от процессов управления на ней. Ее можно разделить на два вида:

метаданные (данные о данных);

собственно условно-постоянную информацию - информацию, подлежащую обработке.

К условно-постоянной информации относятся:

статическая модель станции для системы отображения на цветном графическом терминале;

статическая модель станции для функциональной подсистемы;

технологические номера сигналов, стрелок, секций;

относительные числовые адреса сигналов, стрелок, секций.

Переменная информация зависит от процессов управления и изменяется в реальном времени, например, тексты вводимых директив, очередь системных сообщений, машинная копия полей экрана.

Анализ разработанного ХИИТом и ГТСС макета МПЦ позвололил сделать вывод о том, что от привязки к конкретному объекту зависят не сами программы, а их информационные составляющие, что подтверждает правильность предложенных определений ПО.

'Стратегия проектирования программного обеспечения МПЦ, исходя из предложенной в главе 2 стратегии для гомогенных объектов и классификации ПО, должна предусматривать возможность проектирования ПО как единого целостного программного изделия и включать следующие принципы разработки технологии проектирования ПО:

все управляющие алгоритмы при проектировании прикладного ПО строятся таким образом, что

логическая часть программы и структура ее информационной составляющей для всех объектов данного класса едины,

все частные особенности отдельных объектов управления определяют только объем и содержание (но не структуру) информационной части программы;

прикладное программирование сводится к описанию особенностей конкретного объекта управления;

информационная составляющая прикладного ПО генерируется автоматически.

Исходя из выше изложенного и анализа существующих систем электрической централизации можно сделать следующие выводы: возможность создания единого, функционально-избыточного программного обеспечения МПЦ;

информационной базой для создания ПО конкретной станции должна служить информационная модель станции, позволяющая автоматизированно настроить ПО МПЦ на данную станцию;

создание ПО МПЦ конкретной станции производится путем генерации рабочей версии управляющей ЭВМ из существующего набора модулей (отлаженных один раз) по входным параметрам станции (рис.22);

Процесс генерации рабочей версии управляющей программы ЭВМ

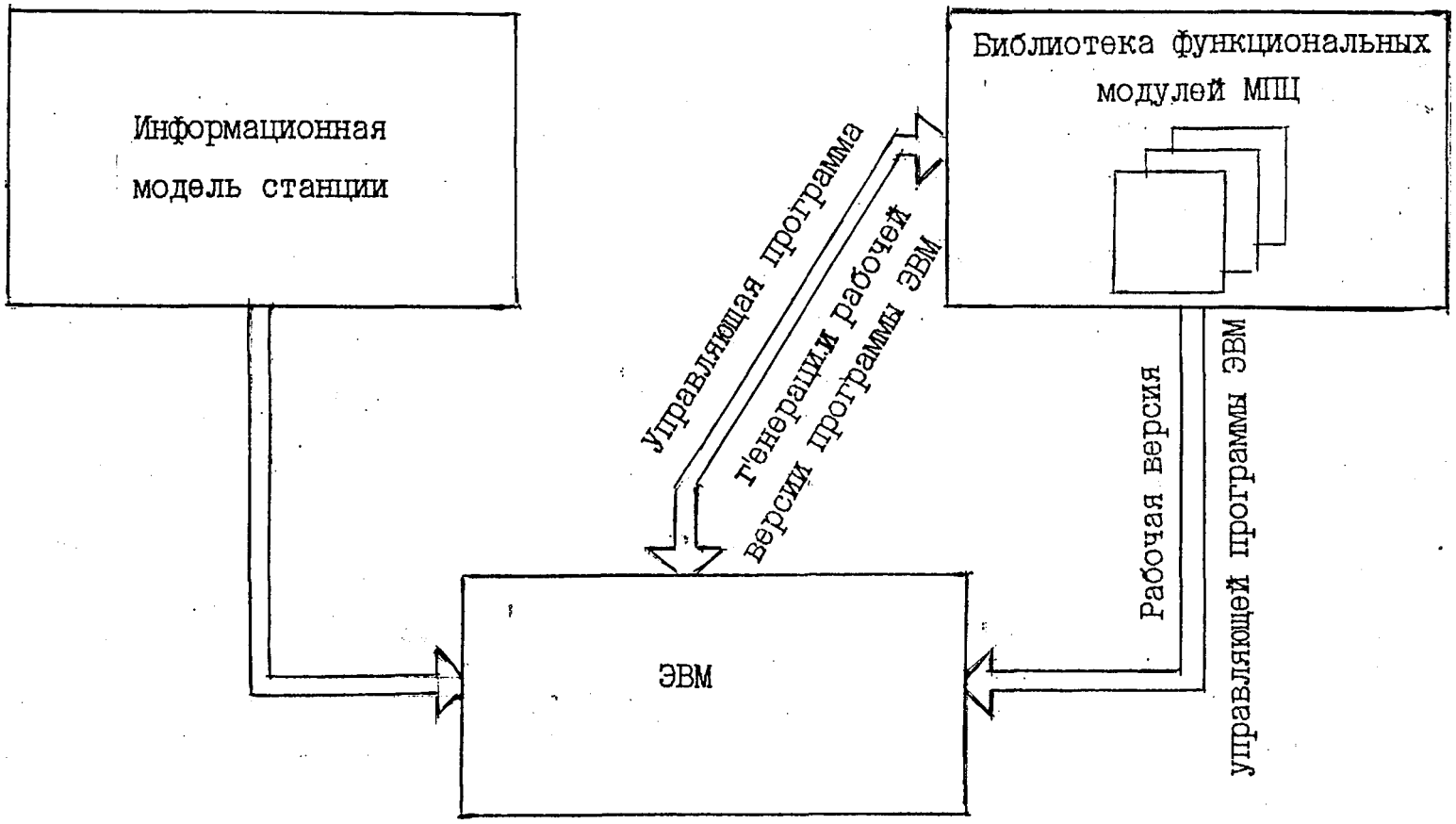


Рис.22

следует разбить объекты управления на три класса (малые, большие, средние). Это позволит на этапе генерации рабочей версии управляющей программы получить количественные и качественные характеристики технических и программных средств для МПЦ конкретной станции. . В частности, на малых станциях слишком дорого иметь полный набор технических средств.

4.6. Генерация программного обеспечения МПЦ

Предложенная стратегия проектирования дает возможность выяснить содержание принципов генерации ПО МПЦ, основанных на применении методов минимизации условно-постоянной информационной составляющей модулей, сведении ее к постоянной составляющей. Среди них:

1. Предусмотреть создание абсолютной библиотеки программных модулей, из которой можно, генерировать тексты для конкретной станции. Для этого использовать принципы функциональной избыточности и избирательности (глава 3). Использовать появляющуюся специфическую информацию для подстройки на конкретный объект. 'Единую, ориентированную на множество станций, технологию проектирования для МПЦ целесообразно синтезировать на базе принципов функциональной избыточности и функциональной избирательности. Организованное в соответствии с этими принципами ПО представляет собой библиотеку программных модулей (некоторую абсолютную библиотеку), из которой можно генерировать текст ПО для конкретной станции, что позволяет освободиться’ от многих повторяющихся этапов его разработки и учесть специфику объекта.

2. Решение лингвистических проблем описания объекта. ***Для*** обеспечения минимальной связности модулей и их максимальной прочности, перемещаемости программ, для унификации информационной базы необходимо создать единую структуру информации. Для этого: использовать на кодовом уровне понятие метаданных (данных о данных), применяя косвенную адресацию; ввести ограничители на длины всех информационных массивов либо счетчики для ограничения на их длину, что позволяет настраивать программы на размеры информационных массивов, а соответственно, ***и*** использовать их для всего множества различных объектов. Все перечисленные методы использованы в разработанных автором функциональных модулях установки маршрута, отмены маршрута, искусственной разделки маршрута [1201.

3. Использовать современные методы проектирования, начиная с внешних спецификаций. В частности, на модульном и кодовом уровнях для достижения максимальной надежности и снижения стоимости разрабатываемого ПО необходимо объединить приемы структурного программирования с методологией проектирования архитектуры, включая бригаду главного программиста, проектирование сверху-вниз, библиотеки, поддерживающие процесс развития проекта и тому подобное.

4. Ввести обязательный контроль достоверности ПО по методу, предложенному в главе 3.

В качестве иллюстрации использования принципов функциональной избыточности и функциональной избирательности и подтверждения необходимости разбивки всех станций на три класса (малые, средние, большие)был проведен анализ способов построения таблицы враждебностей (таблицы возможных

маршрутов).

Для каждой станции существует таблица зависимости, где указываются все возможные маршруты на станции. Был проведен анализ байтового и битового способов построения и использования таблицы зависимости с целью выявления оптимального способа ее хранения с точки зрения объема, занимаемой программой и собственно таблицей памяти.

Результаты проведенного анализа представлены графиком зависимости объема, занимаемой программой памяти от размера станции (рис.23). Оказалось, что для каждого конкретного объекта целесообразно использовать различные варианты построения таблицы. Для малых станций (с количеством светофоров 5 и ниже) наиболее приемлемым является байтовый вариант, для больших - битовый. Причем, с ростом размеров станции выигрыш битового варианта становится весьма существенным. Так, для станции с размером в 10 светофоров разрыв в объеме занимаемой программой памяти составляет 57 байт, а для крупной станции в 40 светофоров разрыв составляет уже 1347 байт. Анализ проведен для выявления оптимального способа хранения таблицы враждебностей в зависимости от размеров станции, к которой осуществляется привязка.

В перспективе предполагается автоматизация наиболее рутинных процессов в проектировании и, в первую очередь, таких как создание наборов данных, отражающих особенности конкретной станции, что позволяет повысить такие конструктивные особенности ПО как корректность на стадии разработки и надежность на стадии эксплуатации.

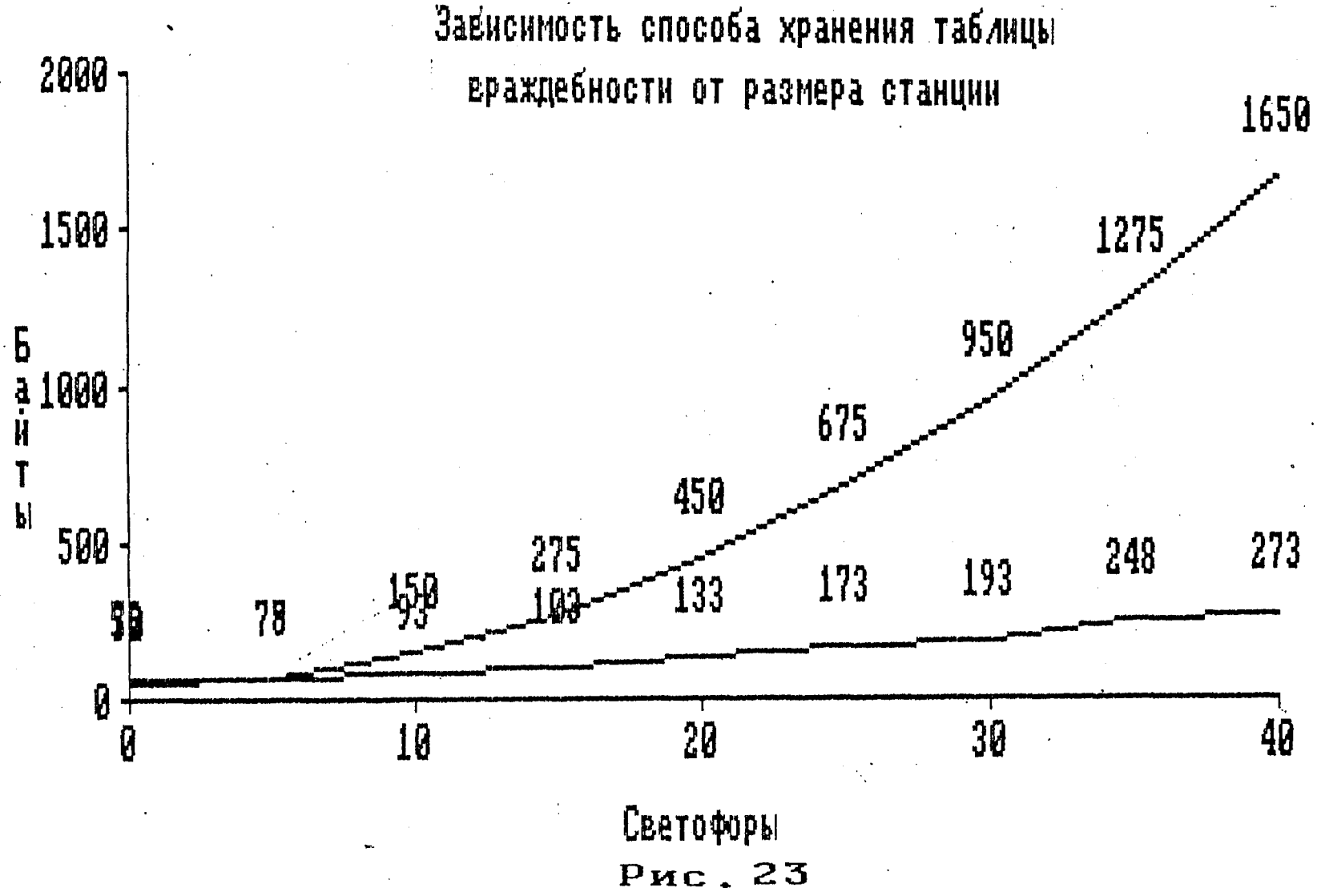

Зависимость способа хранения таблицы
враждебности от размера станции
2000
1500
1000
500
0
Байты
1650
1275
950
675
450
275
150
78
0
10
20
30
40
Светофоры

Рис. 23

Далее глава посвящена проблеме генерации средствами автоматизированной системы проектирования условно-постоянной информации в ПО МПЦ, которая отражает только особенности конкретной станции, но не изменяется в процессе эксплуатации.

4.7. Анализ спецификаций и структуры программного обеспечения МПЦ

Любая система, использующая в качестве, центрального, органа вычислительную систему, требует разработки определенных программных средств. Эти программные средства представляют собой совокупность программных модулей с различной степенью связности между ними как функциональной, так и по данным.

Такую совокупность, разработанную в интересах проектируемой системы, независимо от внешних форм и физических сред хранения и интерпретации, и будем называть ее программным обеспечением .

Под ПО МПЦ следует понимать комплекс программ, физической средой хранения и интерпретации которых является аппаратура МПЦ и, таким образом, в ПО МПЦ не включаются средства автоматизации проектирования, отладки и документирования. Другими словами, под ПО МПЦ понимается такой конечный продукт проектирования, как программное изделие, поставляемое на полигон вместе с аппаратурой.

При решении сложных задач (а задача управления стрелками и сигналами таковой, и является) резко повышается объем и сложность программного- обеспечения. Разработка таких сложных программных проектов без соответствующей методики проектиро-

вания эффективной быть не может. В первую очередь, такая методика должна предусматривать возможность разработки ПО как единого концептуального целостного программного изделия коллективом работников.

Поставленная цель в настоящее время может быть решена в соответствии с технологиями модульного программирования сверху вниз. Суть их состоит в том, что, все ПО разбивается на самостоятельные модули, имеющие минимальные связи друг с другом. То есть модули должны быть в максимальной степени независимы и удовлетворять определению программы . Вместе с тем отдельно взятый модуль, рассматриваемый как самостоятельная программа, работает в среде, создаваемой некоторым другим модулем. Таким образом, между модулями существует отношение подчиненности и в целом модульная структура ПО МПЦ может быть представлена в виде нисходящей иерархической структуры (рис.24).

Разработанная нисходящая модульная структура ПО МПЦ основана на разработанных ГТСС и предыдущем опыте ХИИТа по проектированию МПЦ. Детализация схемы и доведение ее до уровня нулевого приближения может быть произведено после формулировки целей ПО МПЦ и разработки внешних спецификаций.

Все ПО МПЦ строится по модульному принципу. По функциональному назначению всю совокупность модулей ПО МПЦ можно разделить на две контрастные группы:

группа модулей, осуществляющих непосредственно функции управления (установка маршрута, отмена маршрута, включение пригласительного огня и тому подобное);

Архитектура ПО МПЦ

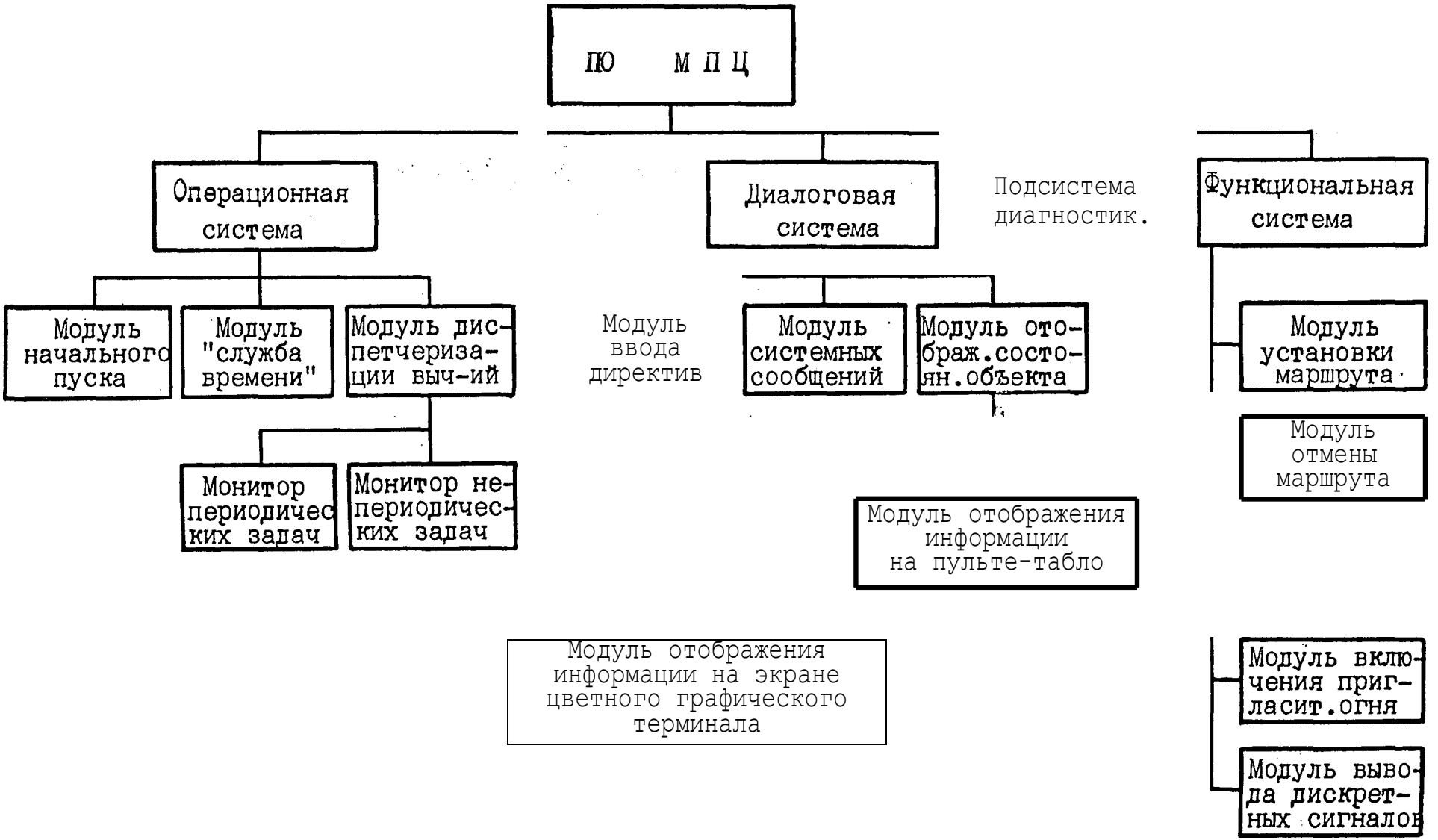


Рис. 24

го
о

группа модулей, осуществляющая общесистемные функции МПЦ (диалоговая система, система отображения информации, модули управления вводом-выводом и ходом вычислительного процесса, программные средства диагностики, повышения надежности и тому подобное).

Первую группу модулей будем называть функциональной системой или прикладным ПО МПЦ. Вторую - системным ПО МПЦ. Системное ПО МПЦ включает в себя модули, непосредственно не связанные с процессом управления стрелками и сигналами, но создающими операционную среду, необходимую для эффективности работы прикладного ПО.

Системное ПО МПЦ включает в себя:

операционную систему;

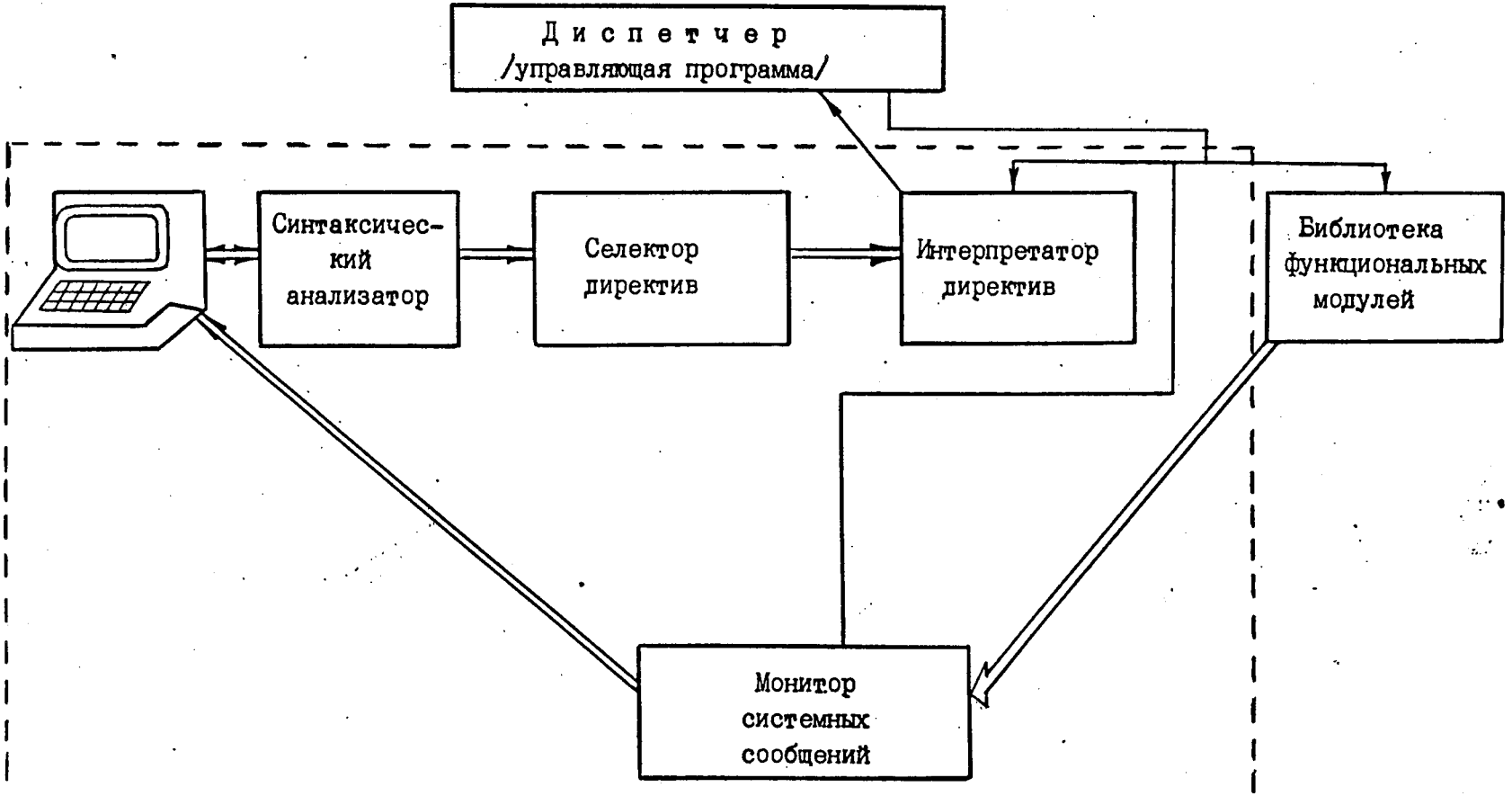
диалоговую систему и систему отображения информации;

программные средства диагностики состояния аппаратных средств и повышения надежности.

4.8. Диалоговая подсистема МПЦ

Понятие гомогенности как абстрактной характеристики ОУ .практически отображается в ПО СУ подобными объектами, поэтому концепция гомогенности применима и к нему. В дальнейшем понятие гомогенности будет применяться без особых оговорок к программному обеспечению систем управления объектами, относящимися к классу гомогенных, и к отдельным его частям, образованным при декомпозиции ПО на составляющие. Так к типу гомогенных можно отнести и диалоговую подсистему МПЦ (рис.25).

Подсистема оперативного диалога



п одсистема оперативного диалога

Рис.25

В ней алгоритмы монитора системных сообщений, синтаксического контроля директив, коды об ошибках, массивы кодов директив и их числовых эквивалентов не зависят от конкретной станции, то есть являются постояннымми. Таким образом, можно предположить, что эти алгоритмы будут разрабатываться один раз.

Однако, статическая модель станции, отображаемая на цветном графическом терминале, технологические номера стрелок, рельсовых цепей, сигналов и их количество (а, следовательно, и размеры соответствующих массивов) для каждой конкретной станции различны. Так как алгоритмы - общие, то эта часть может быть разработана один раз. А информация, имеющая уникальный характер, может генерироваться для каждой станции, то есть она и является объектом автоматизации.

Таким образом, процесс диалога будет сводиться к генерации условно-постоянной информации, определяющей особенности всей информации.

В МПЦ, как замкнутой системе ''объект-регулятор", хотя часть контуров управления и является автоматической, роль регулятора, осуществляющего общую стратегию управления, отведена человеку и, таким образом, в целом, система управления стрелками и сигналами является эргатической. Это обстоятельство превращает ее эргономические характеристики в один из основных (наряду с надежностью) показателей эффективности и качества.

Эргономические характеристики в данном случае определяются не только средствами отображения информации о состоянии объекта управления, но и лингвистической основой диалога

человека—оператора и МПЦ.

Функционально диалоговая система предназначена для ввода человеком в систему управляющих директив и реализации его реакции на ответные сообщения системы (рис.26). С помощью экранного пульта диспетчер может анализировать текущее состояние системы МПЦ, устанавливать требуемые режимы работы, контролировать аварийные ситуации. Это стало возможным благодаря использованию монохроматических и, информационно более насыщенных, полихроматических экранов.

Структурно диалоговая система состоит из модуля ввода директив, монитора системных сообщений, модуля отображения состояния объекта (рис.24).

4.9. Система отображения информации

Подсистема отображения является, по существу, с точки зрения обмена информацией, пассивной. Управление режимами отображения осуществляется с помощью специальных директив диалоговой системы.

Для отображения топологии станции и динамики происходящих процессов могут использоваться цветные графические терминалы (общего и детального вида). Для малых станций в качестве системы отображения целесообразно использовать экран дисплея оперативного диалога. Для крупных (узловых, сортировочных и так далее) станций очевидно есть смысл использовать существующие сейчас на железной дороге пульты управления, согласовав их работу с управляющим вычислительным комплексом ,

Информационные связи в диалоговой подсистеме

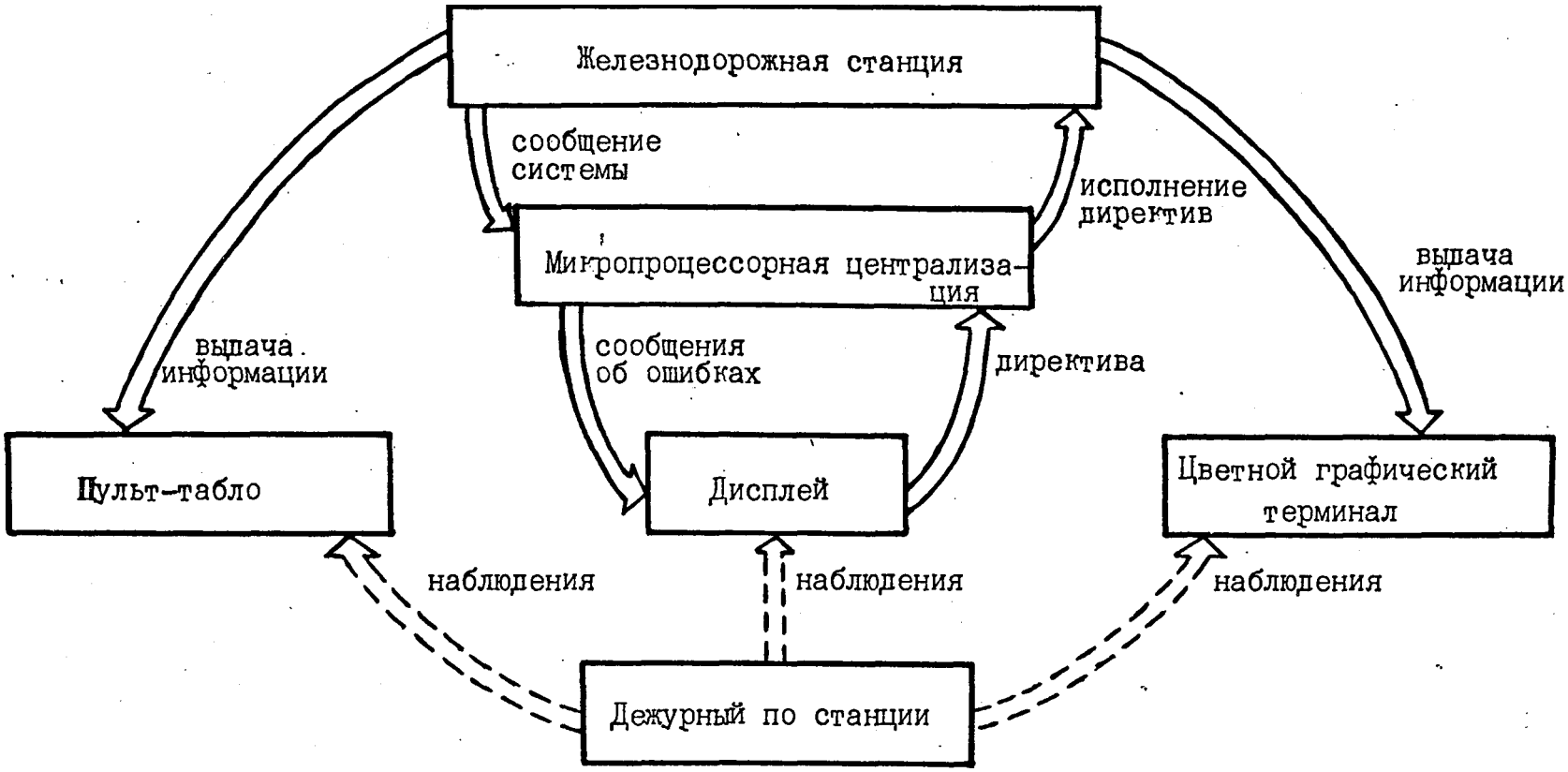


Рис. 26

Задачей подсистемы отображения информации на цветном графическом терминале (ЦГТ) является повышение надежности функционирования системы управления движением поездов в пределах железнодорожной станции на базе МПЦ.

Решение задачи позволит:

предоставить диспетчеру железнодорожной станции структуру объектов управления в традиционном виде;

предоставить текущее состояние объекта управления и его элементов в форме, которая наиболее полно соответствует закономерности восприятия и дальнейшей переработки ее диспетчером железнодорожной станции;

осуществлять оперативную связь с задачами других подсистем, использующих информацию об отображаемых объектах и их элементах.

Информация подсистемы отображения информации на ЦГТ состоит из описаний соответствия физических и логических параметров, состав и структура которых не зависят от конкретной станции. Такими параметрами являются время, вид графического символа, принятого для обозначения объектов и элементов управления, характеристики аппаратной среды (например, размер экрана ЦГТ).

Подсистема отображения информации на пульте-табло является пользователем соответствующей информации, представляемой подсистемой центральных зависимостей, являющейся источником переменной части данных.

Информация подсистемы отображения информации на табло предназначена и для ремонтного персонала и используется им в режиме ремонта и профилактики. Она позволяет ремонтному пер-

соналу оперативно определить место неисправности по информации, выдаваемой подсистемой диагностики.

Принципы построения диалоговой системы были отображены и программно реализованы в разработанном ХИИТом макете МПЦ [44,120].

Изложенный вариант диалоговой системы, являясь опытным образцом, не свободен от недостатков. Его доработка потребует более тщательного учета характеристик выбранной технической базы (в частности, состава и возможностей устройств отображения и ввода), разнообразие условий применения, а, следовательно, и возможных вариантов построения систем МПЦ и других факторов.

4.10. Разработка архитектуры программного обеспечения системы проектирования подсистемы отображения информации

Опыт разработки МПЦ убедительно доказывает, что наиболее трудоемким является процесс описания и ввода в ЭВМ информа-“ции о путевом плане станции, координатах размещения и параметрах напольного и постового технологического оборудования. В частности, в первом разрабатываемом макете МПЦ, ориентированном на станцию Карьерная, были заложены следующие принципы описания станции. Статическая модель станции описывается ориентированным графом, вершинами которого являются рельсовые цепи, а дугами - стрелки, сигналы и их состояния, обеспечивающие переход из одной вершины в другую. Светофоры и стрелки пронумерованы в возрастающем порядке слева направо. Изолированные же участки пронумерованы, исходя из усредненных координат изолированных стрелочных секций. Номера их

присваиваются по мере возрастания их усредненных координат, что необходимо для нахождения маршрута кратчайшей длины. Все описание велось вручную. На этапе поискового проектирования эта система удовлетворяла нас. Первоочередной виделась задача создания функционирующей системы. На нынешнем же этапе надежного проектирования существующие принципы описания объекта:

не могут удовлетворять условиям быстрой кодировки других оъектов (даже для небольших станций необходимо закодировать нулями-единицами порядка 20 светофоров, 10 стрелок, 25 рельсовых цепей, то есть процесс - очень трудоемкий);

источник ошибок очень велик. Контроль оуществлялся, но носил скорей "стихийный" (визуальный) характер. Нет гарантии, что не пропущен тот или иной светофор, рельсовая цепь.

Для создания единого, формального, легко-читаемого описания топологии станции необходимо:

выяснить каковы должны быть средства для описания объекта, в чем состоят правила описания объектов, другими словами, какой должна быть лингвистическая поддержка описания объекта;

создать специальную программу настройки, которая позволяла бы программным путем ставить в соответствие реальные объекты принятой нумерации, то есть иметь формальную закодированную информационную модель, по которой затем осуществляется привязка к конкретной станции;

ввести автоматический контроль вводимой информации;

предусмотреть получение цифровой модели станции с последующим выводом на экран топологии станции для.непосредствен-

ной сверки информационной модели станции с документом-планом станции, что значительно сократит количество ошибок при описании объекта;

в конечном счете добиться автоматического формирования массивов статической модели станции;

использовать появляющуюся специфическую информацию для подстройки на конкретный объект.

Успешное решение этих задач конечным пользователем непрограммистом (инженером-проектировщиком) требует создания языка проектирования путевых объектов - своего рода АРМа-проектировщика описания станции. Цель его создания - получение эффективного средства для ввода в ЭВМ и корректировки описания топологических характеристик и параметров путевых объектов. В конечном счете средствами языка можно предусмотреть формирование моделей путевого плана не .только станций, а и других подобных объектов - перегонов, сортировочных горок. Интерпретация этих объектов средствами машинной графики решает одну из трудоемких задач МПЦ, состоящую в получении графической документации на путевые планы объектов.

Языком здесь может быть либо

специально созданный язык проектирования путевых объектов;

перечень формальных правил описания построения объекта.

Автор использовал второй путь,, используя возможности современных пакетов прикладных программ и средств машинной графики.

Основным документом при проектировании МПЦ является план станции с нанесенными на него элементами: рельсовыми

цепями, светофорами и стрелками.' Опираясь на эти исходные данные, можно построить таблицы зависимостей положения стрелок и сигналов (таблицы враждебности маршрутов), которые для каждой станции являются индивидуальными. Однако при построении этих таблиц разработчики опираются на общие правила. Такие же правила имеются и при разработке плана станции на экране терминала.

Для создания подсистемы отображения информации необходимо иметь самостоятельные модули, обеспечивающие процесс - проектирования статической модели станции:

1. Модуль формирования и обслуживания баз данных графических элементов.

2. Модуль формирования статической модели.

3. Модуль контроля статической модели.

Архитектура подсистемы отображения информации представлена на рис.27.

Построение плана станции на экране терминала осуществляется при помощи специальной программы, исходными данными для которой являются помещенные в память машины в виде файла данных сведения обо всех элементах топологии станции. Это позволяет не создавать специальных программ для каждой станции, а, имея одну общую, создавать план станции-на дисплее и в таком виде поставлять на станцию как готовый программный продукт.

Архитектура ПО подсистемы отображения информации

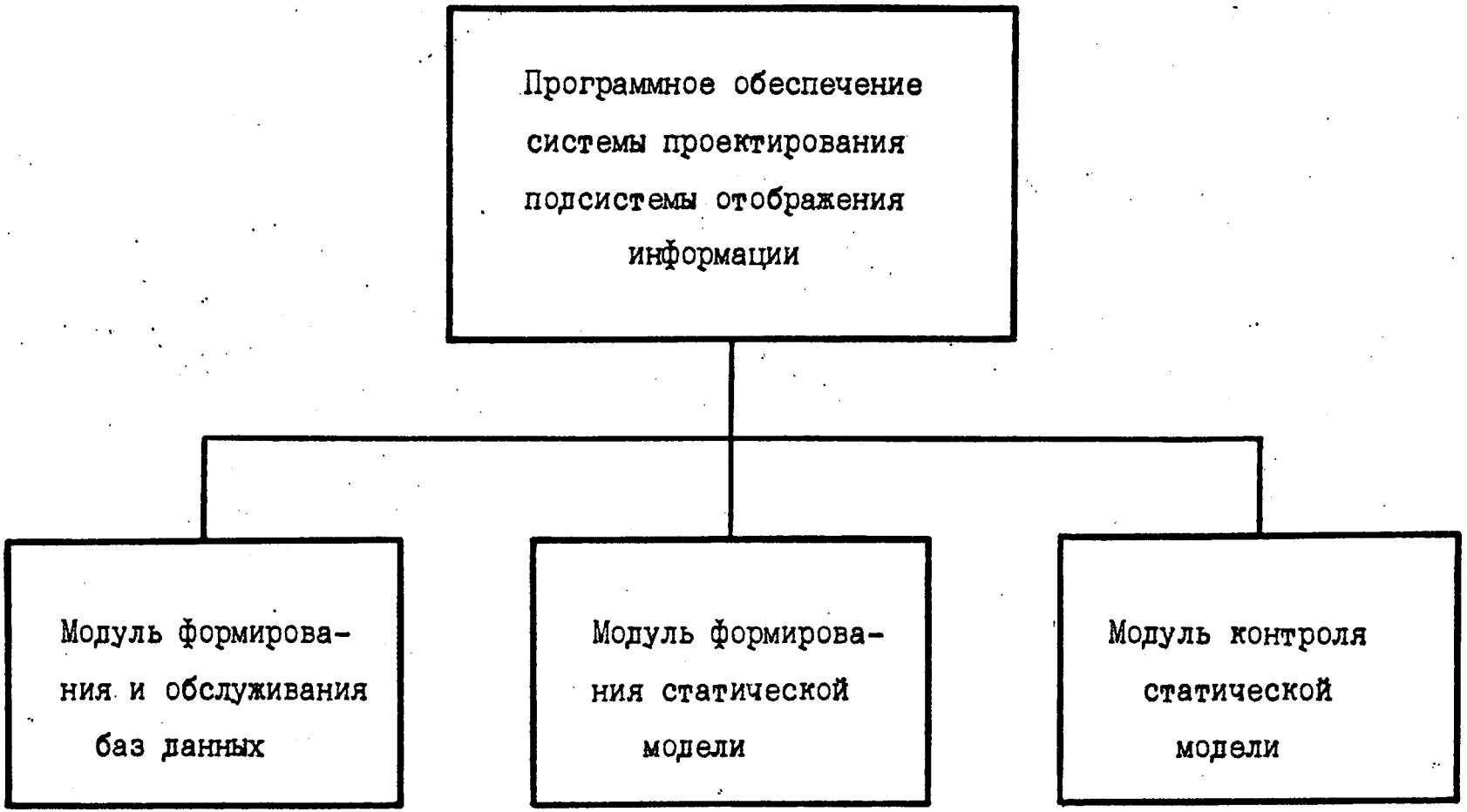


Рис. 27

4.11. Создание и обслуживание баз данных

Технологические этапы проектирования программного обеспечения подсистемы создания и обслуживания баз данных графических представлений элементов железнодорожных станций отображены на рис.9.

4.11.1. Требования и цели программного обеспечения подсистемы создания и обслуживания баз данных

При проектировании программного обеспечения подсистемы создания и обслуживания баз данных конечным продуктом подсистемы являются базы данных, содержащие графическое представление элементов железнодорожных станций. Созданный набор графических изображений может быть использован в подсистеме формирования статической модели станции, как элементная база. „

При создании баз данных необходимо, чтобы. программное обеспечение удовлетворяло следующим требованиям:

1. Система должна работать в диалоговом режиме.

2. Должна предоставлять пользователю комфортные условия.

3. Элементы, входящие в состав базы, должны изображаться на полихроматическом дисплее.

4. Система должна обеспечить все функции, необходимые для ведения базы данных:

4.1. Возможность добавления новых элементов.

4.2. Редактирование имеющихся элементов.

4.3. Удаление элементов.

4.4. Поиск элемента в базе.

4.5. Просмотр элементов.

5. Система должна обеспечивать возможность вывода изображения, хранящегося в базе данных в любом масштабе, необходимом пользователю.

6. Ввиду того, что пользователь не является профессионалом в области вычислительной техники, система должна быть легко устанавливаемая и простая в обращении.

7. Работа системы должна обеспечивать высокую культуру обслуживания пользователей всех уровней.

Для проектирования программного обеспечения подсистемы создания и обслуживания базы данных, исходя из выше перечисленных требований, можно сформулировать следующие цели:

8. Данная подсистема предназначена для создания и ведения баз данных, содержащих графическое представление элементов железнодорожных станций, которая поставляется на конкретную станцию вместе с другими элементами микропроцессорной централизации и является базовой для работы других подсистем МПЦ, например, подсистемы формирования статической модели станции. Подсистема обслуживания баз данных позволяет автоматизировать проектирование МПЦ..

9. Подсистема должна позволять, при создании баз данных, осуществлять ввод, корректировку, удаление и поиск данных.

10. Система должна быть совместима по своим данным с подсистемой формирования статической модели.

11. Созданная подсистемой база данных должна быть защищена таким образом, чтобы пользователь не мог внести в нее изменения случайным образом.

4.11.2. Выбор технических и программных средств

При разработке ПО подсистемы создания и обслуживания баз данных для МПЦ необходимо учитывать, на какой аппаратуре будет работать ПО.

В качестве технических средств выбраны ПЭВМ фирмы IBM и совместимые с ней, как наиболее широко распространенные. Эта ПЭВМ должна отвечать следующим требованиям: центральный процессор любой из 80286/80386/80486 с математическим сопроцессором, объем ОЗУ не менее 640 Кб, накопитель на жестком диске с объемом не менее 10 Мбайт, не менее одного накопителя на гибком магнитном диске любого объема, разрешающая способность полихроматического дисплея не менее 640 x 350 точек, клавиатура и принтер стандартные для IBM. Кроме того, ПЭВМ должна содержать операционную систему типа MS DOS V.3.3 и выше.

Для создания системы создания и обслуживания баз данных элементов железнодорожных станций использовалась СУБД общего назначения Clipper’87. Важное достоинство системы Clipper’s? - возможность использования расширения оперативной памяти персонального компьютера при использовании разработанных его средствами приложений.

В данной диссертационной работе используется пакет dGE и весьма развитый пакет PCX Programmer’s Toolkit фирмы Gems Microprogramming.

Библиотека dGE (database Graphics Extension) предоставляет возможность работы с графикой для программ на языках dBase - совместимых систем.

Значительно более богатыми функциями располагает пакет PCX. Он позволяет создавать прикладные программы на традиционных языках программирования и языках СУБД. Пакет включает библиотеку, содержащую более 60 графических функций, а также девять программ-утилит. Утилиты позволяют показать изображение из файла на экране, поместить изображение с экрана в файл, отпечатать изображение на матричном или лазерном принтере, создать фрагмент изображения, преобразовать текстовый экран в формат PCX, управлять библиотекой изображений.

4.11.3. Внешнее проектирование

1. Предварительный внешний проект

Основными элементами подсистемы обслуживания и создания баз данных графических представлений элементов железнодорожных станций являются непосредственно базы данных, хранящие математическое описание графических элементов.

При формировании и обслуживании баз данных основными функциями пользователя являются следующие:

1. Изучение инструкции пользователя для работы по формированию и обслуживанию баз.

2. Возможность создания графических элементов.

3. Редактирование ранее введенных элементов.

4. Операции над базами данных (поиск, просмотр, дополнение, удаление).

5. Возможность использования сервисных программ.

6. Выход из программы.

2. Детальный внешний проект

Ввиду того, что детальный внешний проект, по существу, представляет собой инструкцию пользователю, автор считает нецелесообразным освещение данного раздела в предлагаемой работе. Основные результаты работы программного продукта приведены в приложении 2.

4.11.4. Проектирование структуры программы

Процесс проектирования структуры программы включает определение всех модулей программы, их иерархии и сопряжения между ними.

Для того, чтобы достичь минимальной сложности структуры программы, следует придерживаться некоторых критериев.

Основные из них:

1. Модули должны обладать минимальной связностью друг с другом, то есть должны быть независимы друг от друга.

2. Внутри модуля все его элементы должны быть связаны друг с другом как можно более сильно, то есть модули должны быть прочными.

В результате внешнего проектирования были определены основные функции проектируемой программы. Поэтому из проведенных ранее этапов проектирования можно получить глобальную структуру программы, представленную на рис.28. На рисунке каждая функция, описание которой приведено ниже в детальном внешнем проекте, может быть представлена при программной ее реализации в виде модуля, организующего ее выполнение. Эти модули являются абсолютно независимыми друг от друга, и, в свою очередь, могут быть разбиты на простые модули более низкого уровня, реализующие элементы основных функций.

Структура программы формирования и обслуживания баз данных элементов железнодорожной станции

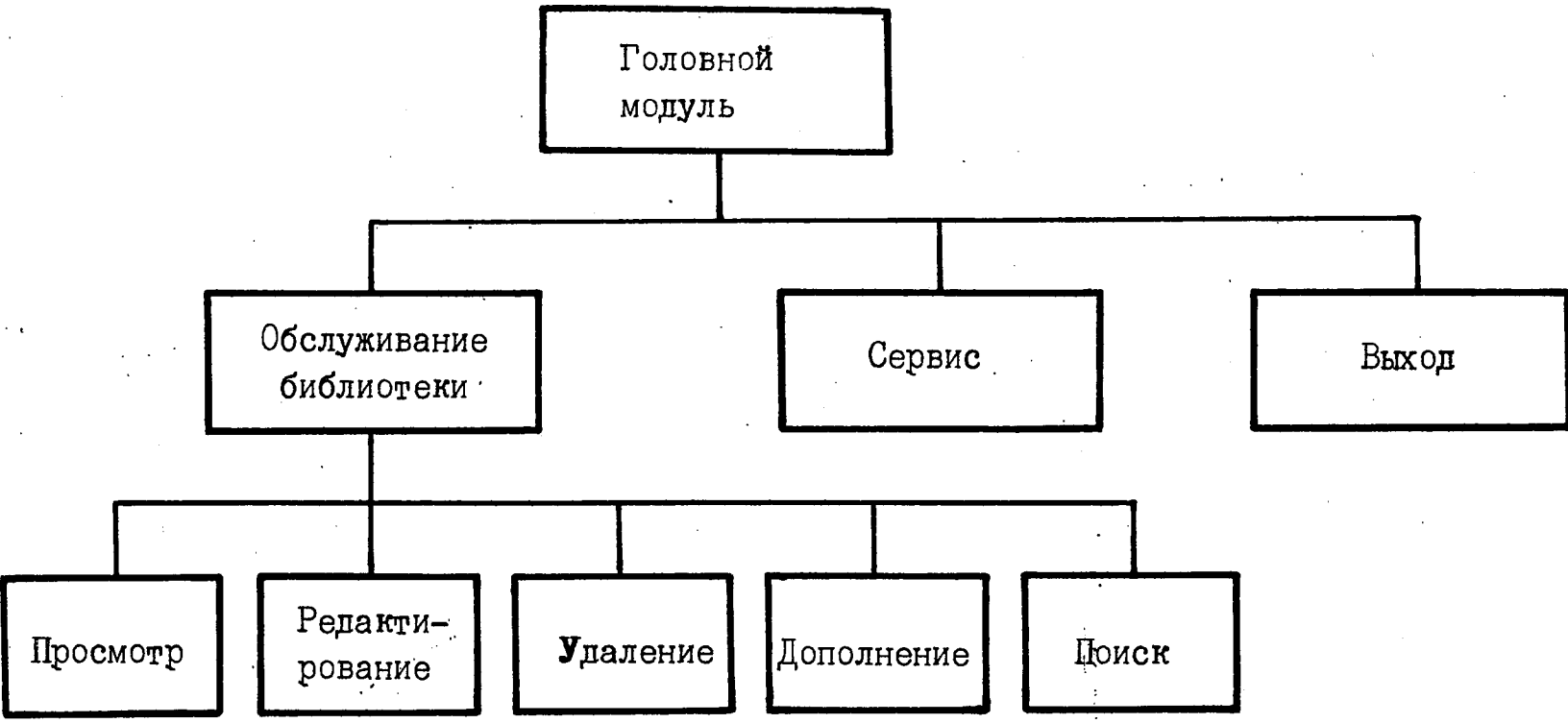


Рис. 28

со

В глобальной структуре основным модулем самого верхнего в иерархии уровня является головной модуль, осуществляющий взаимодейсвие между функциями (то есть их вызов) и работу с пользователем, что позволяет самим модулям-функциям быть независимыми друг от друга. Из рис. видно, что ни один модульфункция пользователя не может быть вызван из другого модуля-функции, находящегося на одной иерархической ступени с ним, а вызвать модуль-функцию может только головной модуль всей программы. Это позволяет проектировщику, во-первых, проектировать -модули-функции отдельно и независимо друг от друга (это проектирование может осуществляться разными людьми, что ускоряет процесс создания программы); во-вторых, тестирование и отладка каждого модуля-функции может вестись автономно до того момента, когда эти функции будут подключены к головному модулю; в третьих, при внесении изменений в какой-либо модуль-функцию они не отражаются на других функциях. Все эти модули будут иметь максимальную прочность и минимальную связность, что крайне желательно при построении надежных программ.

В силу этих причин при проектировании структуры программы и в дальнейшем будем использовать технологию модульного программирования. Полученная на рис. глобальная структура как раз и отвечает требованиям модульной технологии, так как имеет древовидную структуру. В дальнейшем такую же структуру будут иметь и модули более низких уровней.

Рассмотрим детально структуру программы рис.29. Здесь указаны названия всех■ входящих в программу модулей и их иерархия.

Детальная структура программы IRONLIB

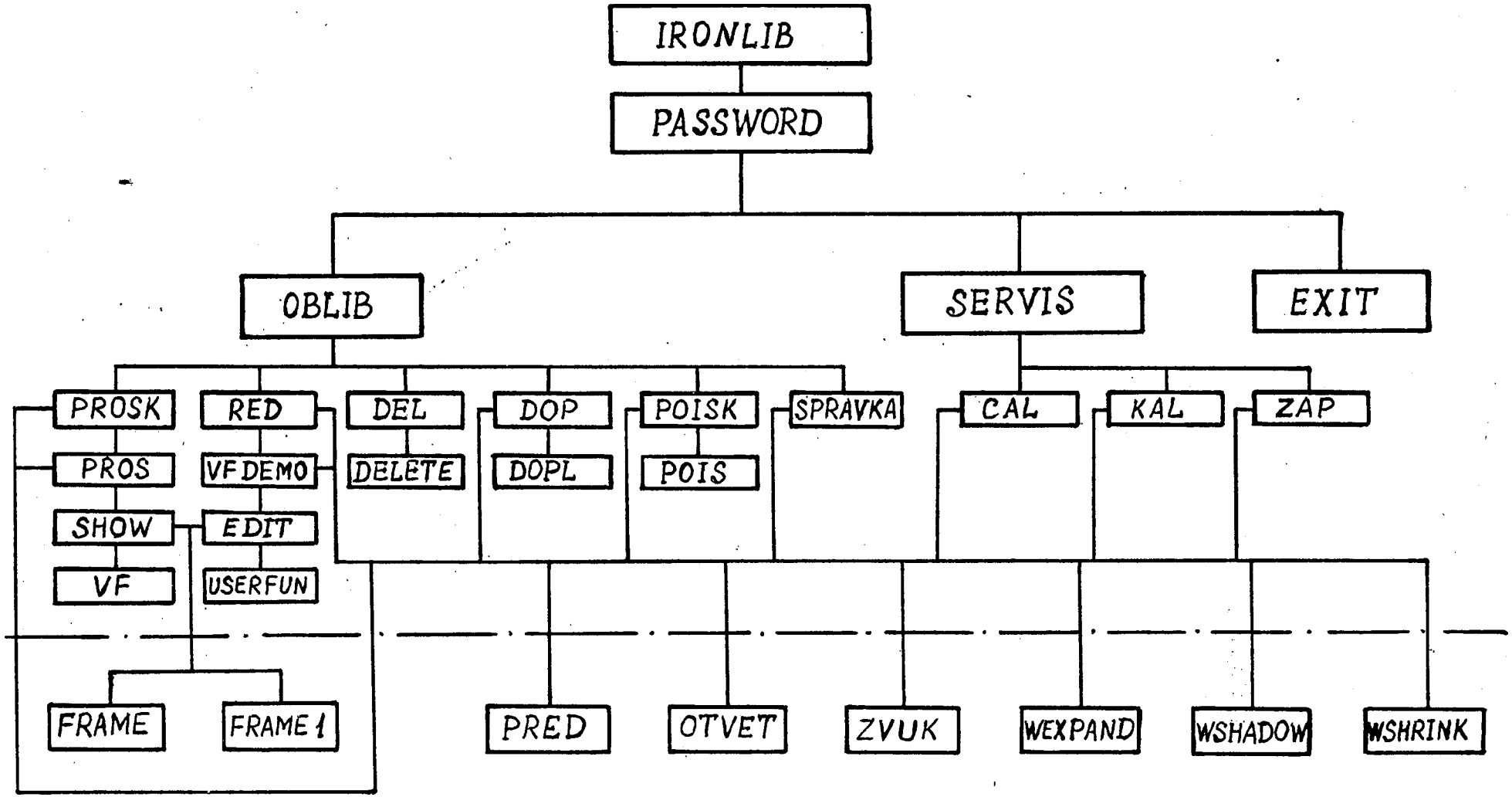


Рис. 29

Головной модуль осуществляет вызов модулей "OBLIB" (обслуживание библиотеки), "SERVIS" (сервис), после введения пользователем пароля, который обеспечивает модуль "PASSWORD". Модуль "OBLIB" в свою очередь обеспечивает вызов модулей "PROSK" (просмотр), "RED" (редактирование), "DEL" (удаление), "DOP" (дополнение), "POISK" (поиск), "SPRAVKA" (справка). Каждый из этих модулей, в свою очередь, осуществляет вызов модулей более низкого уровня, которые выполняют действия непосредственно над базами данных, либо используют модули еще более низкого уровня.

Модуль "SERVIS" осуществляет вызов модулей "CAL" (калькулятор) и "KAL" (календарь).

Таким образом, полученная древовидная структура выполняет все поставленные перед ней функции по обслуживанию баз данных элементов железнодорожных станций. Причем модули, находящиеся на одном уровне иерархии выполняют родственные функции, но никак не связаны между собой и вызываются только модулями, находящимися на более высоком уровне иерархии. Модули, содержащие функции, выполняемые во многих точках программы, значительно сокращают программу за счет того, что они вынесены на самый нижний уровень, так как для соблюдения принципа модульности нам было бы необходимо в каждом месте, где требуется операция, содержащаяся в подпрограммах нижнего уровня, полностью создавать ее вновь. Это могло бы привести к дополнительным ошибкам. Эти подпрограммы являются служебными и их можно рассматривать как операции, расширяющие возможности используемого языка программирования.

4IL5. Внешнее проектирование модулей программы.

Внешнее проектирование начинается с определения внешних характеристик модуля и выражется в виде его внешних спецификаций, которые не содержат ничего другого, кроме сведений о логике модуля или о внутреннем представлении данных.

Головной модуль программы носит имя ''IRONLIB" и осуществляет диалог с пользователем, то есть воспринимает, какую директиву выберет пользователь и передает управление вызванному модулю. Выходными параметрами является директива, выбранная пользователем в главном меню программы, а внешним эффектом модуля - само головное меню и возможность при помощи клавиш управления курсором выбрать в нем необходимую директиву.

Модуль ''PASSWORD” осуществляет контроль пароля. Входным параметром является пароль, вводимый пользователем. Выходным параметром является либо выход из программы, либо запуск головного модуля. Внешне работа модуля проявляется в том, что внизу экрана появляется окно с предложением ввести пароль. Модуль ''OBLIB” вызывается из главного меню после выбора директивы ''обслуживание библиотеки”. Этот модуль предоставляет пользователю возможность выбрать любую из содержащихся в меню функций по обслуживанию баз данных. Выходным параметром модуля является название выбранной пользователем функции. Внешним эффектом - меню для выбора.

Модуль ''PROSK” вызывается из модуля "OBLIB" после выбора пользователем функции ''просмотр”. Основная функция модуля ''PROSK” - это работа с пользователем и передача управления

модулю более низкого уровня. Внешним эффектом этого модуля является меню с представленным в нем списком имеющихся баз данных . Выходным параметром модуля является название выбранной базы.

Модуль "PROS” вызывается из модуля "PROSK". Входным параметром этого модуля является выбранная база данных. Модуль выполняет функции по просмотру элементов находящихся в базе, перемещению по базе, поиску элемента в базе. Внешним эффектом этого модуля является меню, в котором представлены по десять элементов базы.

Модуль ''SHOW” вызывается из модуля "PROS" после выбора элемента для просмотра его графического представления. Входным параметром модуля является название элементая, выбранного для просмотра. Внешним эффектом модуля является экран с изображением элемента. Модуль выполняет функции уменьшения, увеличения элемента, просмотр следующего и предыдущего элементов.

Модуль "V?" используется модулем "SHOW” для изображения элемента. Входным параметром для этого модуля являются координаты векторов, составляющих этот элемент. Внешним эффектом модуля является изображение, элемента.

Модуль ''RED” вызывается из модуля ''OBLIB” после выбора пользователем пункта ''редактирование”. Основная функция модуля ''RED” - предоставление пользователю возможности выбора имеющихся баз данных. Внешним эффектом модуля является меню с перечисленными в нем имеющимися базами данных. Выходным пара метром является название выбранной базы.

Модуль "VEDEMO” вызывается из модуля ''RED”. Выходным

параметром для этого модуля является название выбранной базы данных. Внешне работа модуля проявляется в переходе в графический режим и оформлении экрана редактора.

Модуль ’’EDIT" используется модулем "VEDEMO". Входным параметром модуля является информация о структуре базы данных и количество записей, приходящихся на изображение каждого элемента. Внешним эффектом модуля является вывод базы данных на экран для полноэкранного редактирования. Выходным параметром служит база данных с внесенными в нее изменениями.

Модуль "USERFUN" используется модулем "EDIT" и представляет собой пользовательскую функцию для полноэкранного редактирования базы данных. Входными параметрами для этого модуля являются введенные с клавиатуры (при помощи функциональных клавиш) функции, которые хочет выполнить пользователь. Модуль выполняет функции: помощь, поиск, быстрое перемещение по базе данных, просмотр графического изображения элемента, выход.

Модуль "DEL" вызывается из модуля "OBLIB” после выбора пункта "удаление". Модуль осуществляет возможность выбора пользователем необходимой базы данных. Внешним эффектом модуля является меню с перечисленными в нем имеющимися базами данных. Выходным параметром является название выбранной базы.

Модуль "DELETE" вызывается из модуля "DEL". Входным параметром для модуля служит название выбранной базы данных. Модуль выполняет функцию удаления элемента из базы данных. К входным параметрам модуля относится так же введенное пользо-

вателем название удаляемого элемента. К внешним эффектам модуля относится окно с предложением ввести название элемента. Выходным параметром модуля является база данных с удаленным элементом.

Модуль "DOB'' вызывается из модуля ''OBLIB" после выбора пункта ''дополнение''. Основная функция модуля - предоставить пользователю возможность выбора имеющихся баз данных. Внешним эффектом модуля является меню с перечисленными в нем имеющимися базами данных. Выходным параметром является название выбранной базы.

Модуль ''DOPL" вызывается из модуля ”DOP”. Входными параметрами для модуля служат название выбранной базы данных и в веденное пользователем название элемента. К внешним эффектам модуля относится окно с предложением ввести название элемента. Выходным параметром является база данных с внесенными в нее дополнениями.

Модуль ''POISK” вызывается из модуля ''OBLIB'' после выбора

пункта ''поиск”. Функция модуля - обеспечить выбор пользователем необходимой базы данных. Внешним эффектом модуля является меню с перечисленными базами данных. Выходным параметром является название выбранной базы данных.

Модуль ”POIS” вызывается из модуля "POISK”. Входными параметрами для модуля служат название выбранной базы данных и введенное пользователем название элемента поиска. Внешний эффект модуля выражается в появлении окна с предложением ввести элемент для поиска . Выходным параметром является найденный элемент.

дуль содержит текст, информирующий пользователя о версии пакета и о средствах, при помощи которых он создан. Входным параметром для данного модуля является сам факт вызова этого модуля, а выходным - текст и возврат после его прочтения в модуль "OBLIB”. Внешний эффект модуля - окно с текстом.

Модуль "SERVIS” вызывается из модуля "IR0NLIB”. Модуль осуществляет диалог с пользователем, воспринимает, какой пункт меню выберет пользователь, и передает управление выбранному модулю. Входным параметром является пункт, выбранный пользователем. Внешним эффектом модуля является меню с перечисленными функциями.'

Модули ”GAL" и ”KAL” являются самостоятельными файлами.

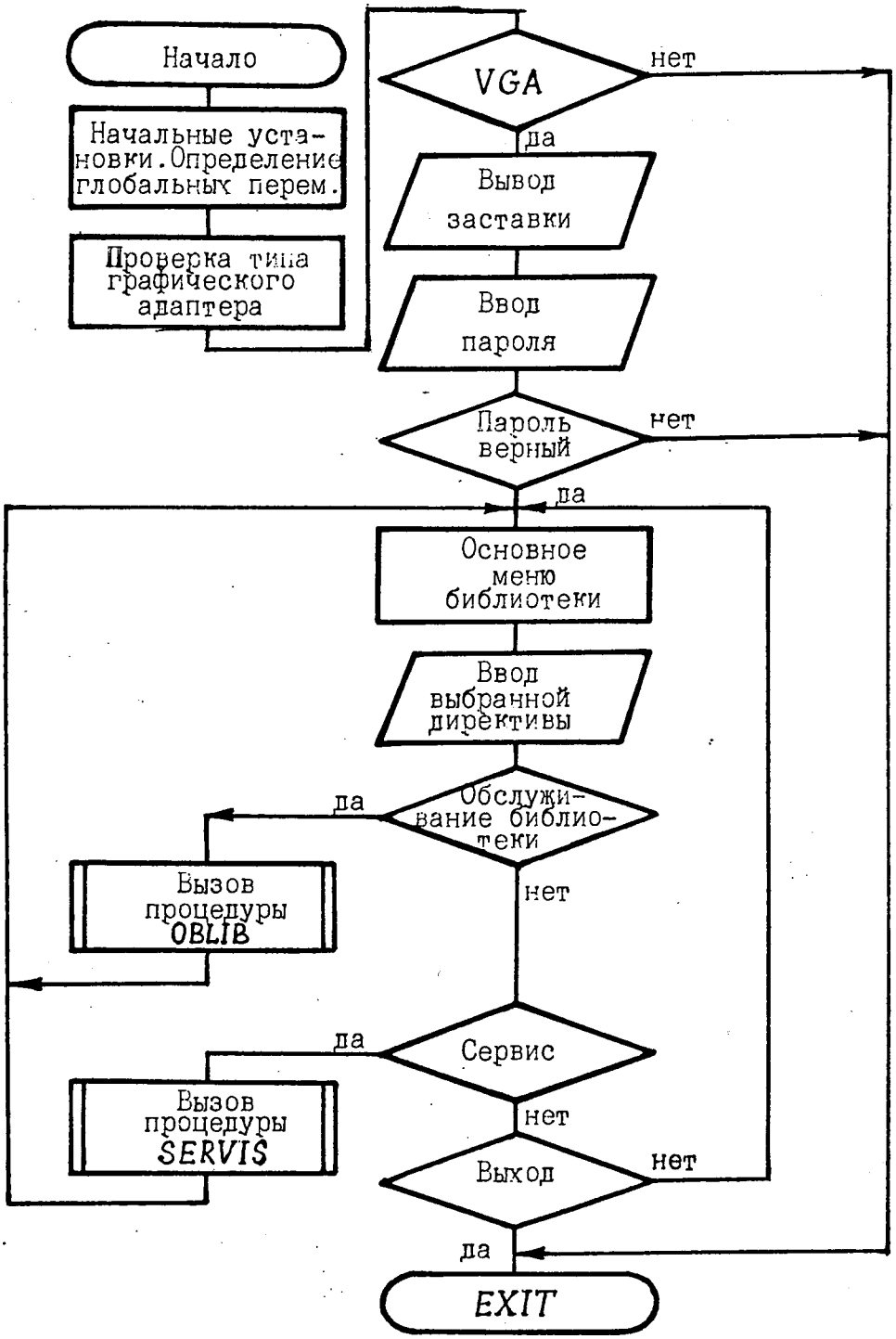
4.11.6. Проектирование логики модулей программы.

Последним в цепи процессов проектирования ПО является процесс проектирования и собственно программирования внутренней логики каждого модуля.

На рис.30 изображен алгоритм работы главного модуля программы "IRONLIB”.

После начала работы главного модуля производятся начальные установки глобальных переменных. Проверяется тип графического адаптера, если на машине установлен не VGA адаптер, то осуществляется выход из программы. Модуль производит проверку знания пароля, если пароль неверный, то осуществляется выход. Затем на экране выводится главное меню программы, в котором перечислены директивы, доступные пользователю. Из этого меню пользователь может выбрать одну из директив и нажать <ENTER>.

Алгоритм формирования и обслуживания баз данных элементов железнодорожной станции



РИСгЗО

Тогда, в зависимости от выбранной директивы, модуль вызовет соответствующий модуль-функцию. Когда модуль-функция закончит свою.работу, программа возвратится в главное меню.

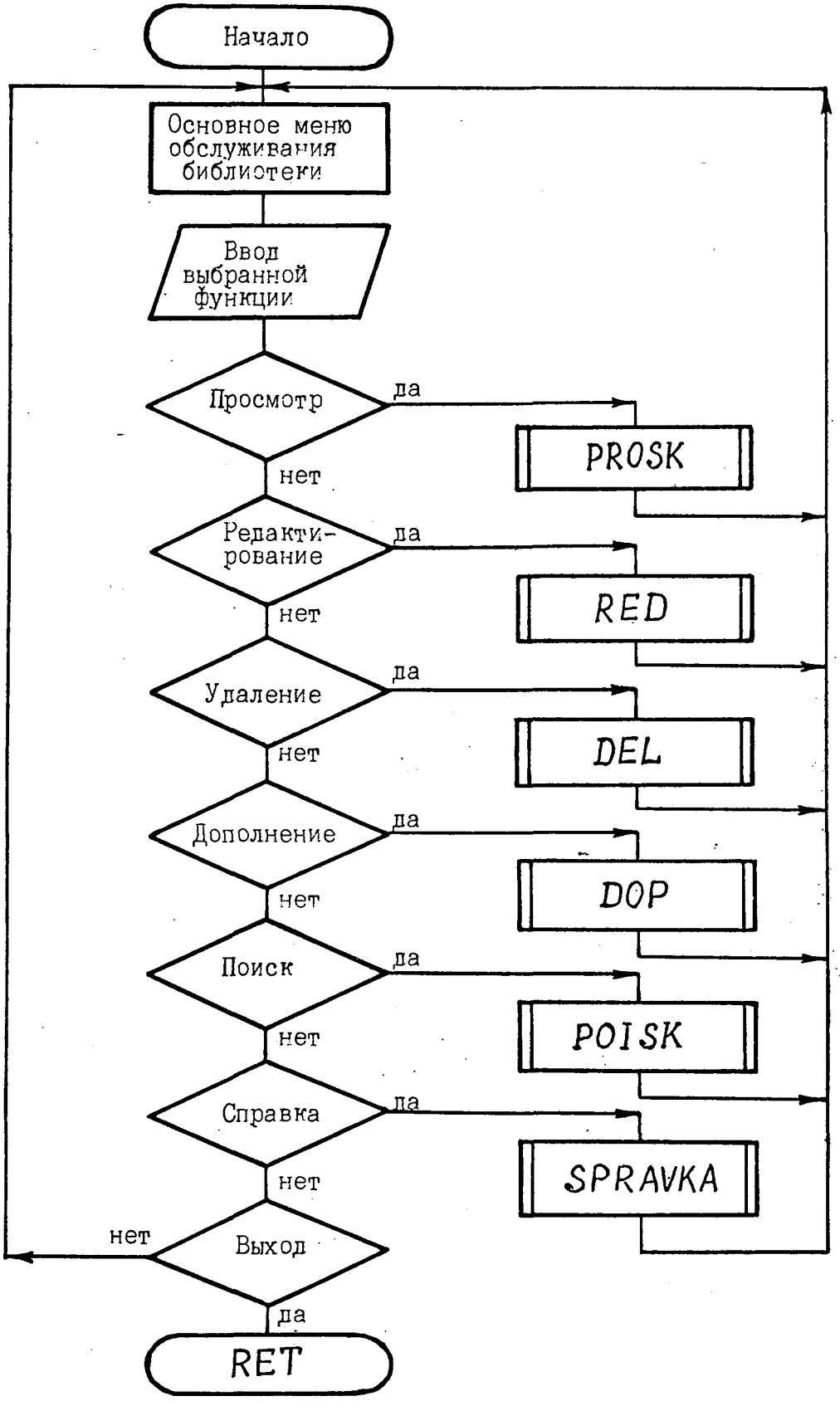
На рис.31 изображен алгоритм работы модуля "OBLIB". После вызова этого модуля выводится меню обслуживания библиотеки. Пользователю предоставляется возможность выбрать из этого меню необходимую функцию. В зависимости от выбранной функции модуль вызовет соответствующий этой функции модуль.

Алгоритм работы модуля "PROSK" представлен на рис.32. Модуль выводит на экран меню с перечисленными в нем базами данных. После выбора пользователем необходимой ему базы данных модуль проверяет наличие такого файла на диске. Если файла не существует, вызывается модуль "PRED" (предупреждение), в случае его наличия происходит, открытие этого файла и вызов модуля "PROS".

На рис.33 изображен алгоритм модуля "PROS". Модуль выводит на экран по десять названий элементов, содержащихся в открытой предшествующим модулем базе данных. Модуль вызывает программы, осуществляющие просмотр элементов базы и выводя графического изображения элемента.

На рис.34 представлен алгоритм модуля "SHOW". Модуль обеспечивает функцию просмотра графического изображения элемента, а также ряд функций, предоставляющих пользователю удобства при просмотре. На экран выводится изображение элемента. В зависимости от выбранной пользователем функции вызывается та или иная подпрограмма, обеспечивающая работу этой функции.

Алгоритм модуля OBLIB



РйС:31

Алгоритм модуля PROSK

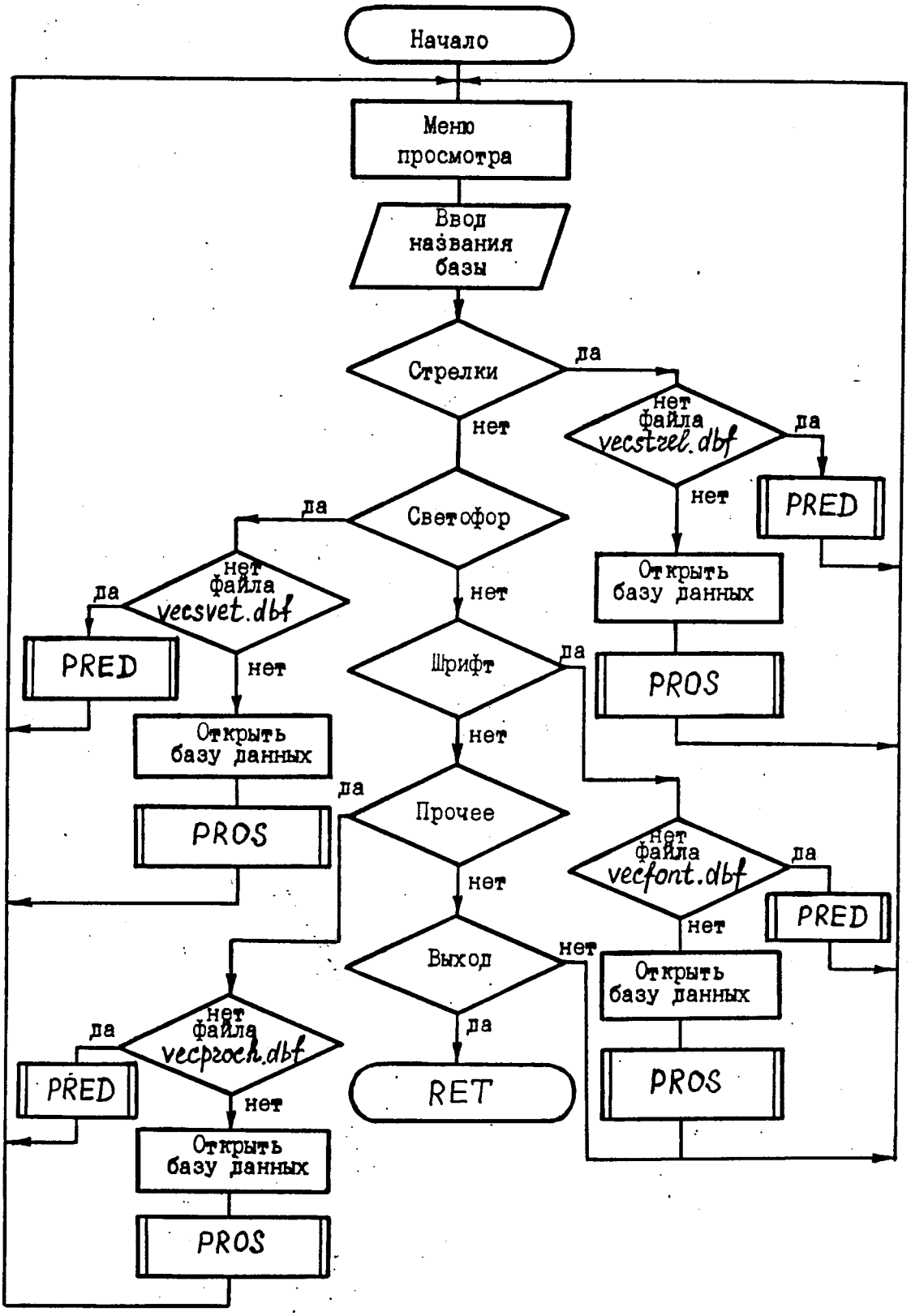


РИС.32

Алгоритм модуля PROS

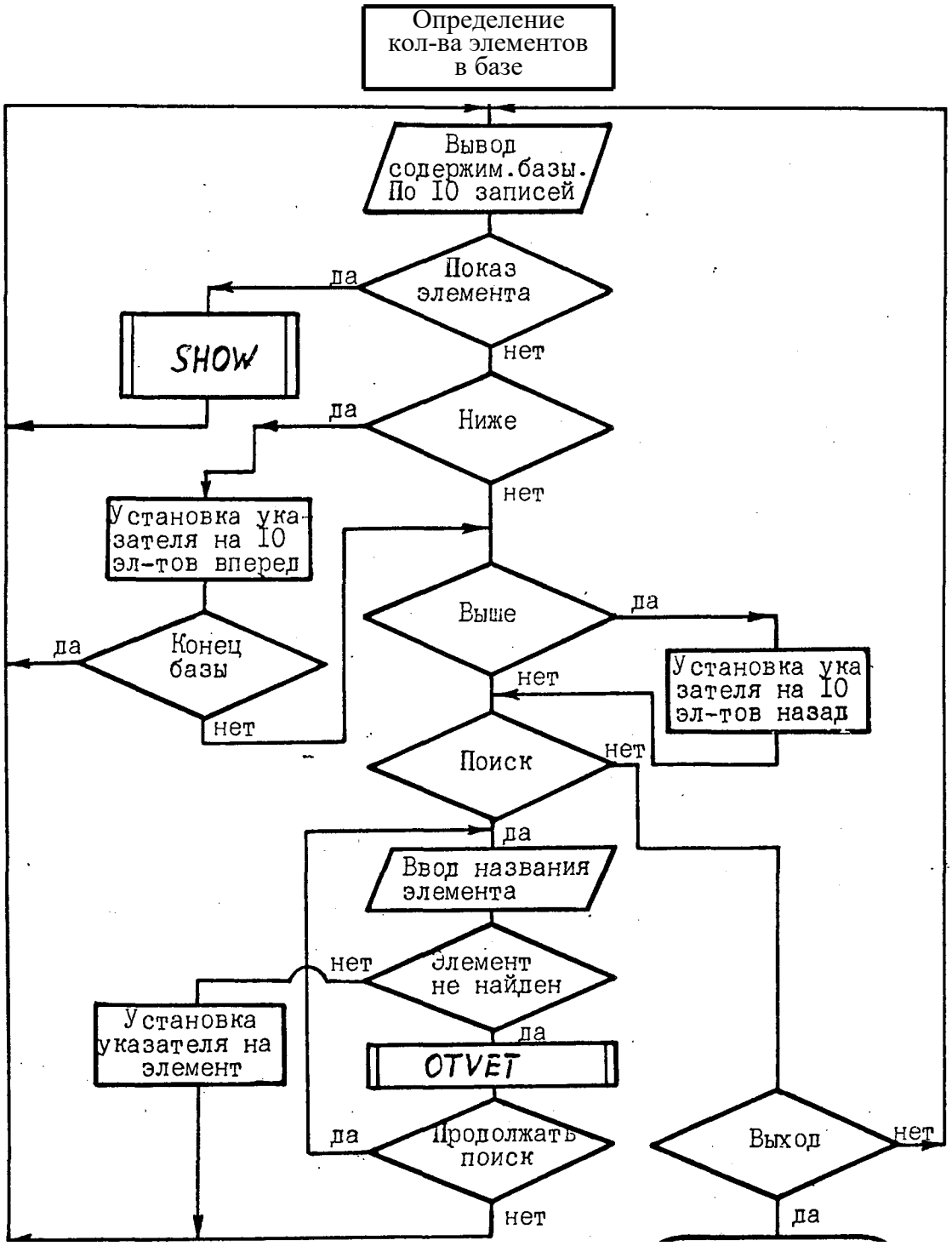


РИС.33

Алгоритм модуля SHOW

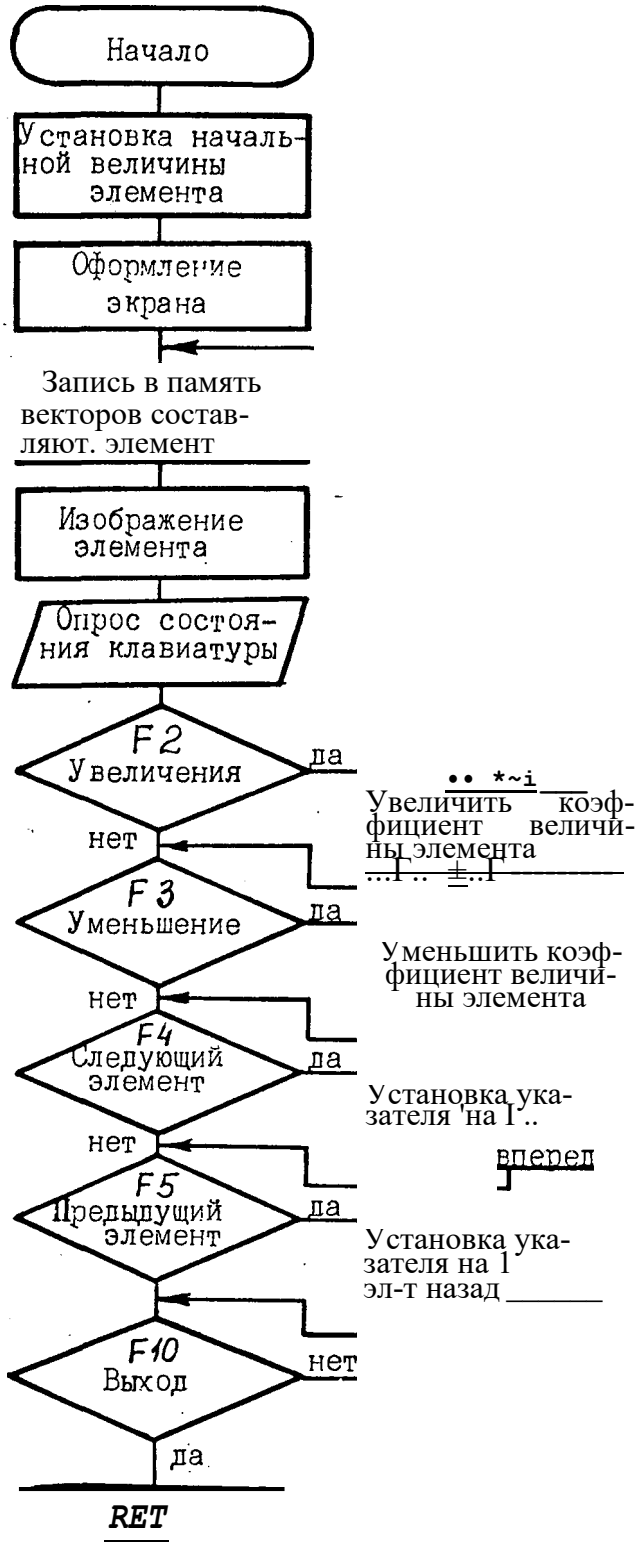


РИС.34

На рис.35 изображен алгоритм модуля ’’RED”. Модуль выводит на экран меню с перечисленными в нем базами данных. После выбора пользователем базы данных модуль проверяет наличие файла на диске. Если файла на диске не существует, вызывается модуль "PRED", в противном случае происходит открытие этого файла и вызов модуля "VEDEMO", который загружает знакогенератор, необходимый в .графическом режиме, оформляет экран и вызывает процедуру "EDIT". После выхода из процедуры "EDIT" модуль устанавливает текстовый режим и передает управление модулю "RED".

Модуль "EDIT" определяет конфигурацию базы данных и осуществляет полноэкранное редактирование базы. Выход из модуля осуществляется в модуль ’’VFDEMO".

Модуль "EDIT" использует для своей работы модуль "USEREШ", алгоритм которого приведен на рис.36 . Модуль осуществляет закрепление за функциональными клавишами определенных функций, производит опрос этих клавиш и, в зависимости от нажатой клавиши, вызывает выполнение той или иной функции.

На рис.37 изображен алгоритм модуля "DEL". Модуль выводит на экран меню с перечисленными ’ в нем базами данных. После выбора пользователем базы данных модуль проверяет наличие такого файла на диске. Если файла на диске не существует, вызывается модуль "PRED”, иначе происходит открытие файла и вызов модуля "DELETE".

Алгоритм модуля "DELETE" представлен на рис.38. Модуль запрашивает пользователя ввести название удаляемого элемента, осуществляет проверку на ввод названия.

Алгоритм модуля RED

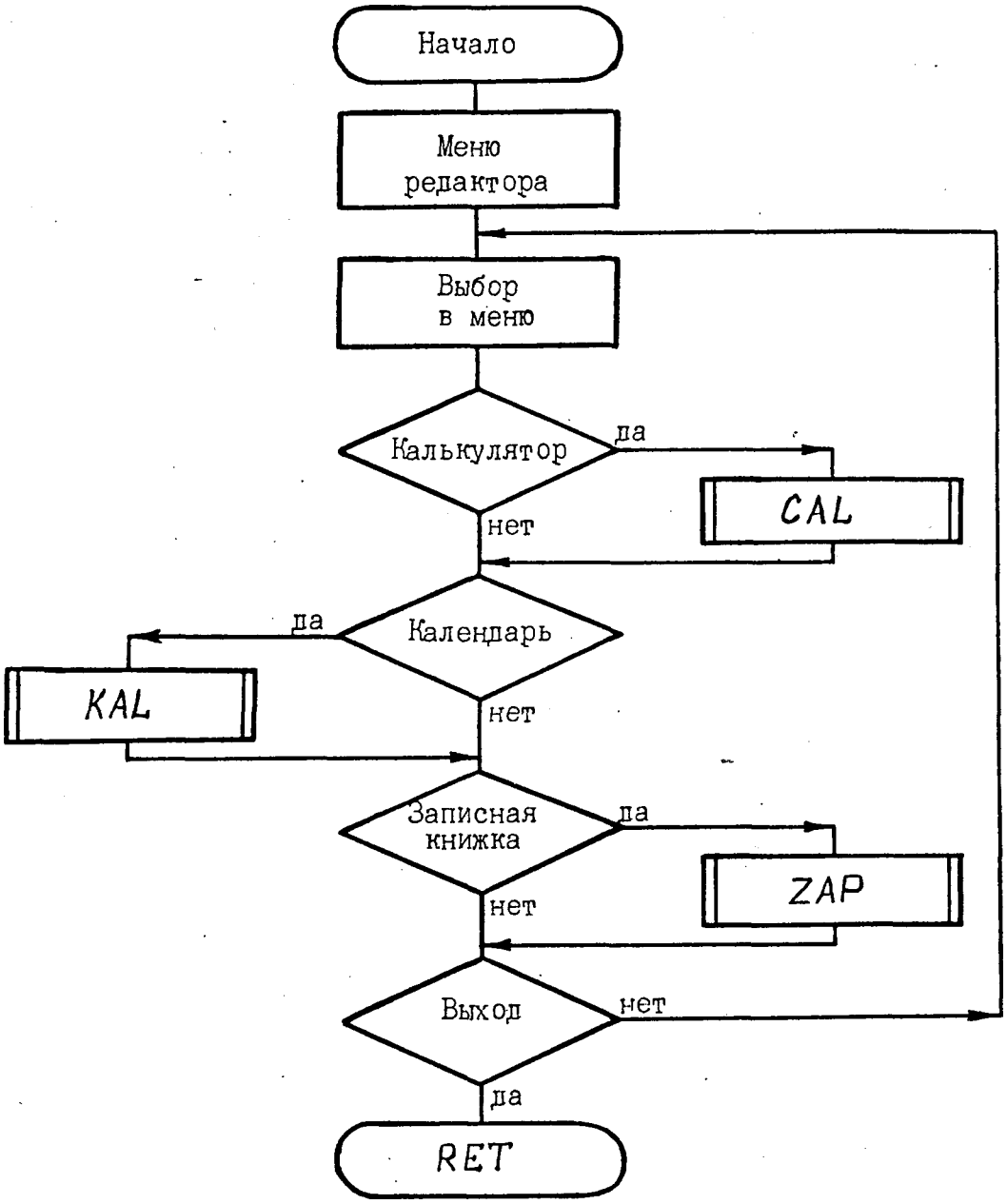
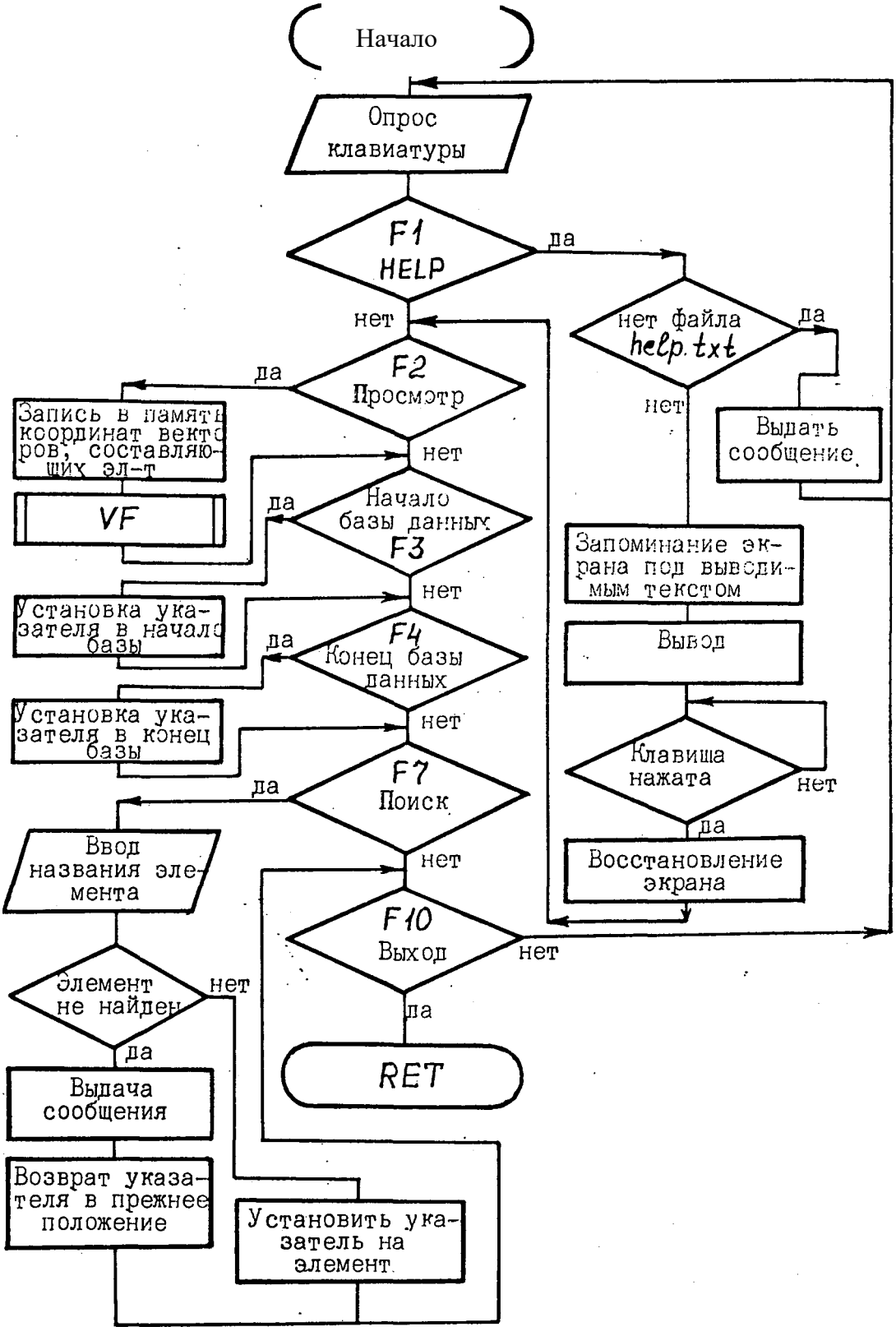


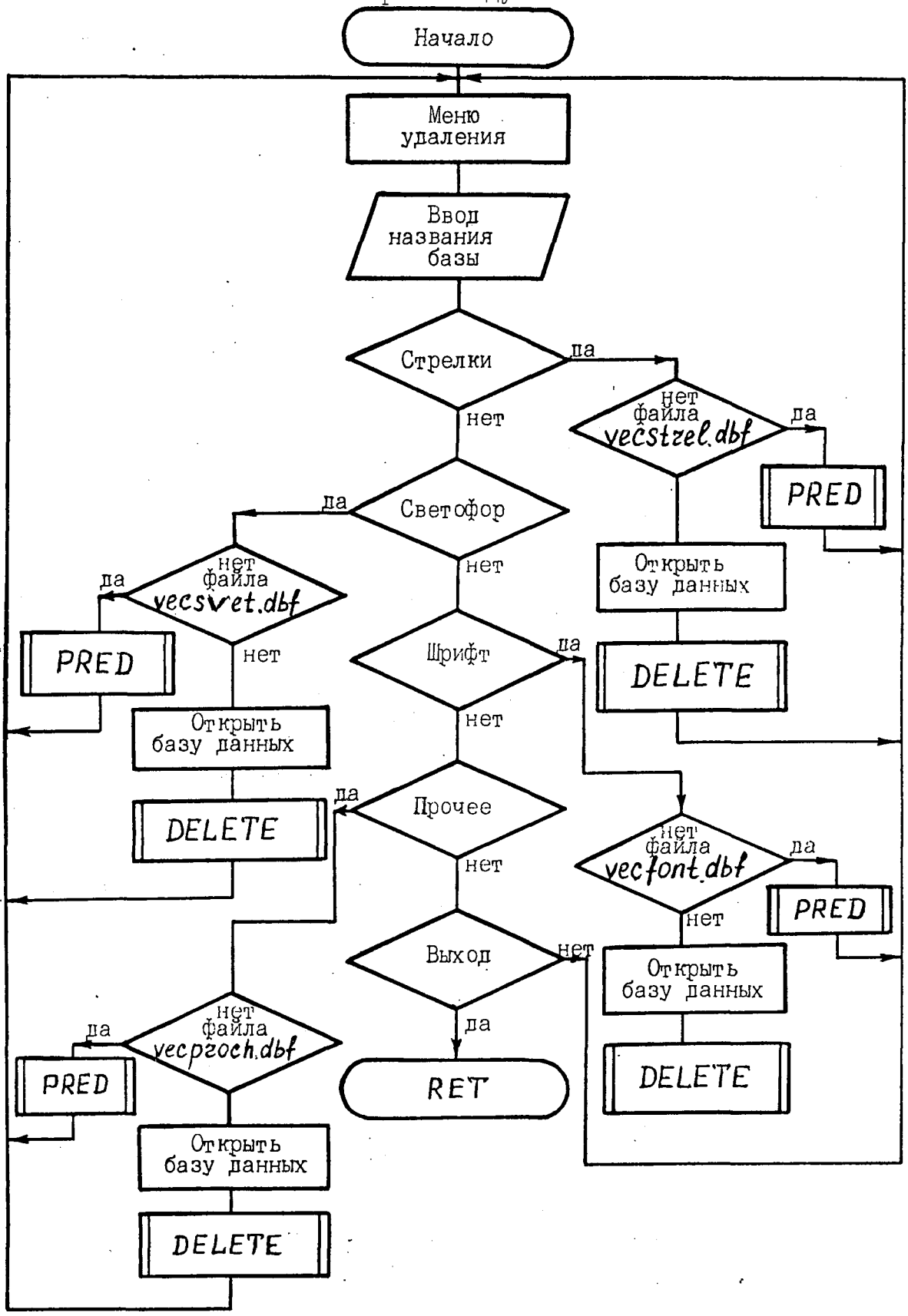
Рис.35

Алгоритм модуля USERFUN



РИС, 3

Алгоритм модуля DEL



Ж, 37

Алгоритм модуля DELETE

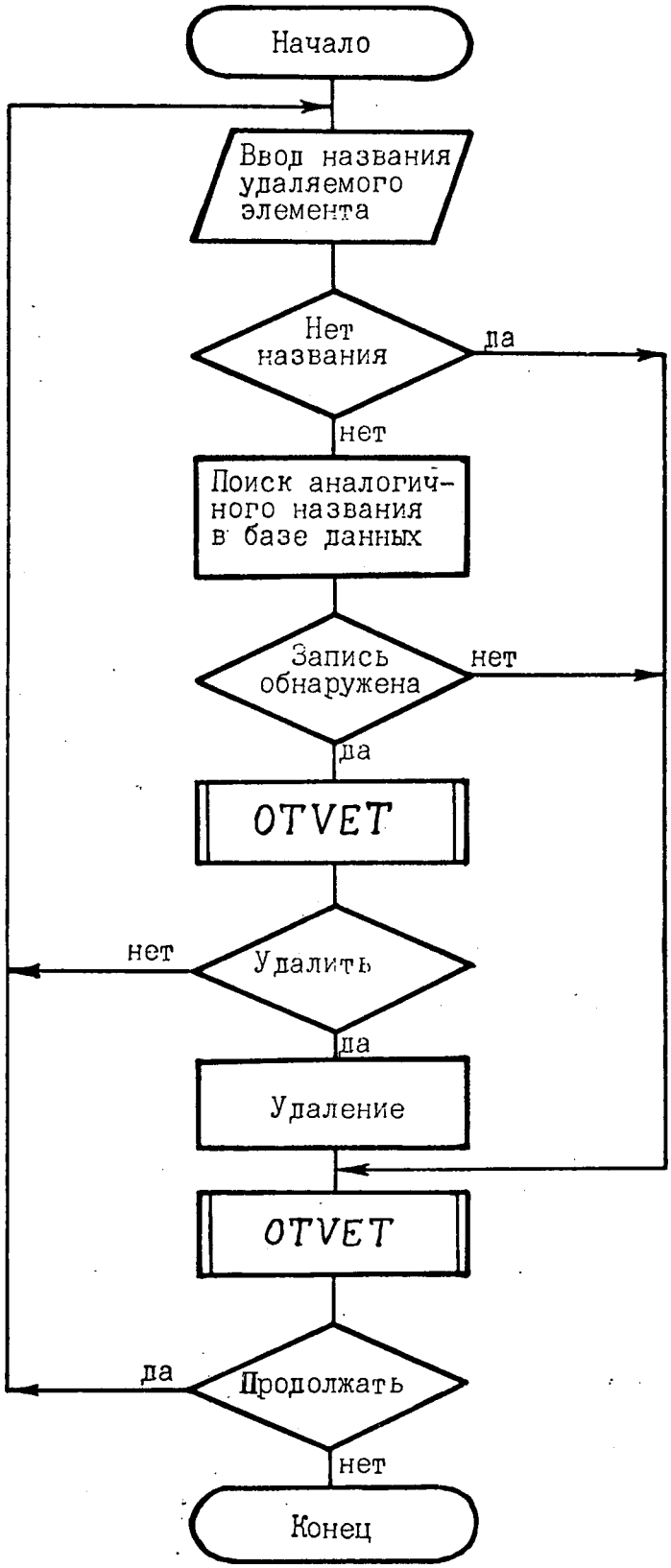


Рис.38

В том случае,если названия не было, модуль предупреждает об этом и запрашивает пользователя о намерении продолжать. После ввода названия модуль ищет этот элемент в базе и удаляет его. Если элемент в базе не обнаружен, система предупреждает об этом и снова запрашивает пользователя о его намерениях. Модуль после окончания своей работы возвращает управление модулю ''DEL".

Модуль "BOP" выводит на экран меню с перечисленными в нем базами данных. Пользователь может выбрать любую из них. После этого модуль проверяет наличие такого файла на диске. Если файла на диске не существует, вызывается модуль "PRED", иначе происходит открытие файла и вызов модуля "DOPL".

Модуль "DOPL" запрашивает пользователя ввести название элемента, а затем осуществляет поиск элемента с аналогичным названием в базе данных. Если элемент обнаружен, модуль добавляет этот элемент в базу данных. Если нет, то выдает об этом сообщение. В обоих случаях модуль запрашивает пользователя о его желании продолжать дополнение. Если "да", то модуль снова начинает свою работу, если ''нет", то модуль передает управление модулю "DOP".

На рис.39 представлен алгоритм модуля '"POISK". Модуль выводит на экран меню с перечисленными в нем базами данных. После выбора пользователем базы данных модуль проверяет наличие такого файла на диске. Если файла на диске не существует, вызывается модуль "PRED", иначе происходит открытие файла и вызов модуля "POPS".

Алгоритм модуля POISK

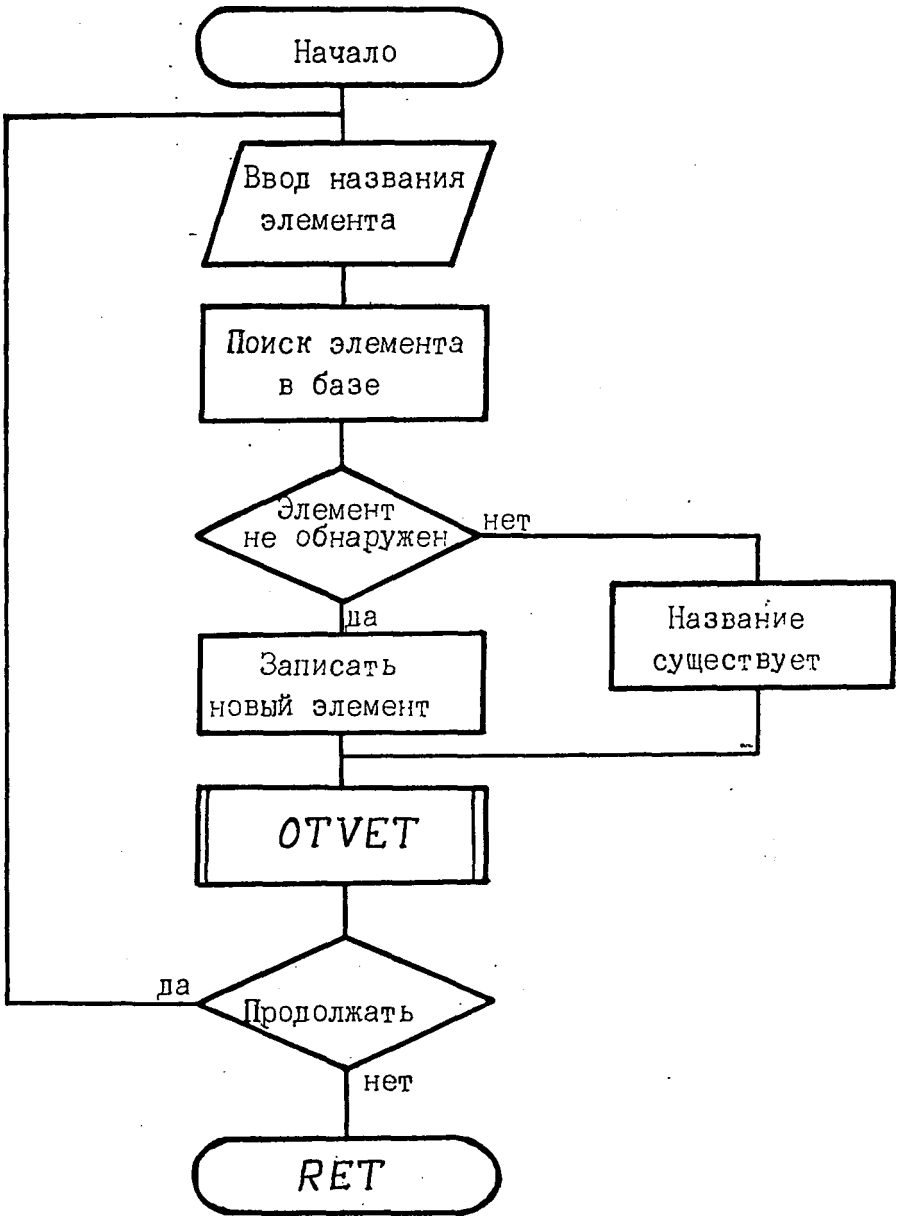


Рис.39

Алгоритм модуля "POIS" приведен на рис.40. Модуль определяет количество элементов, находящихся на данный момент в базе данных. Модуль запрашивает пользователя ввести название элемента для поиска, а затем осуществляет поиск элемента с аналогичным названием в базе данных. Если элемент с таким названием обнаружен, система информирует об этом пользователя, если элемент не обнаружен, система также выдает об- этом информацию. В обоих случаях пользователю задается вопрос оего намерении продолжать. Если - "да", то модуль снова начинает свою работу, если - "нет", то модуль передает управление модулю "POISK".

4.12. Проектирование программного обеспечения подсистемы генерации статической модели станции в МПЦ.

На рис.41 изображен алгоритм работы головного модуля программы "IRONEDIT". После начала работы производятся начальные установки глобальных переменных. Затем на экране выводится главное меню программы, в котором перечислены директивы, доступные пользователю.. Из этого меню пользователь может выбрать одну из функций и нажать <ENTER>. Тогда в зависимости от выбранной функции модуль вызывает соответствующий модуль-функцию. Когда модуль-функция закончит свою работу, программа вернется в главное меню. Текст программы данного модуля приведен в приложении.

Алгоритм модуля POPS

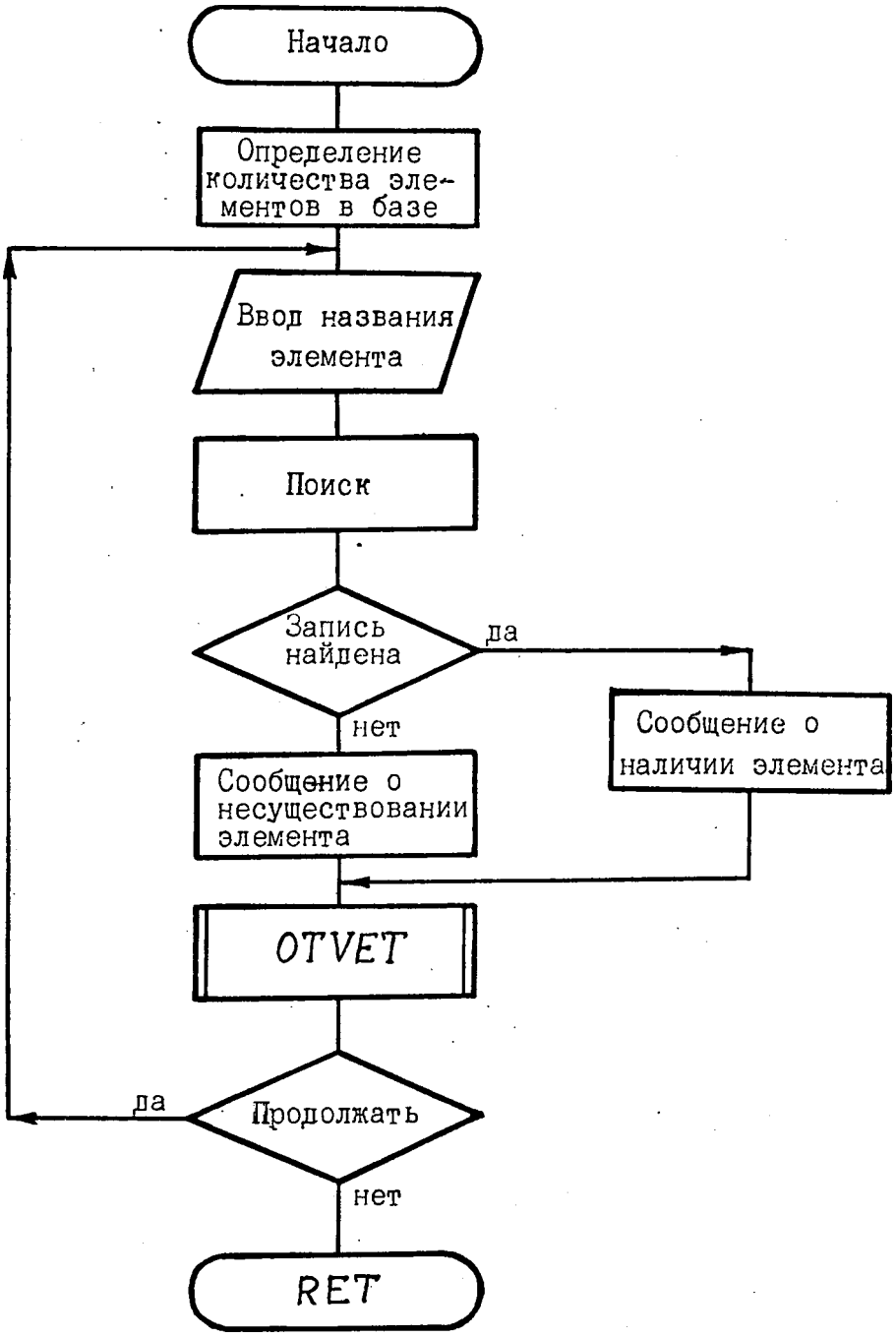


Рис.40

Структура программы генерации статической модели железнодорожной станции

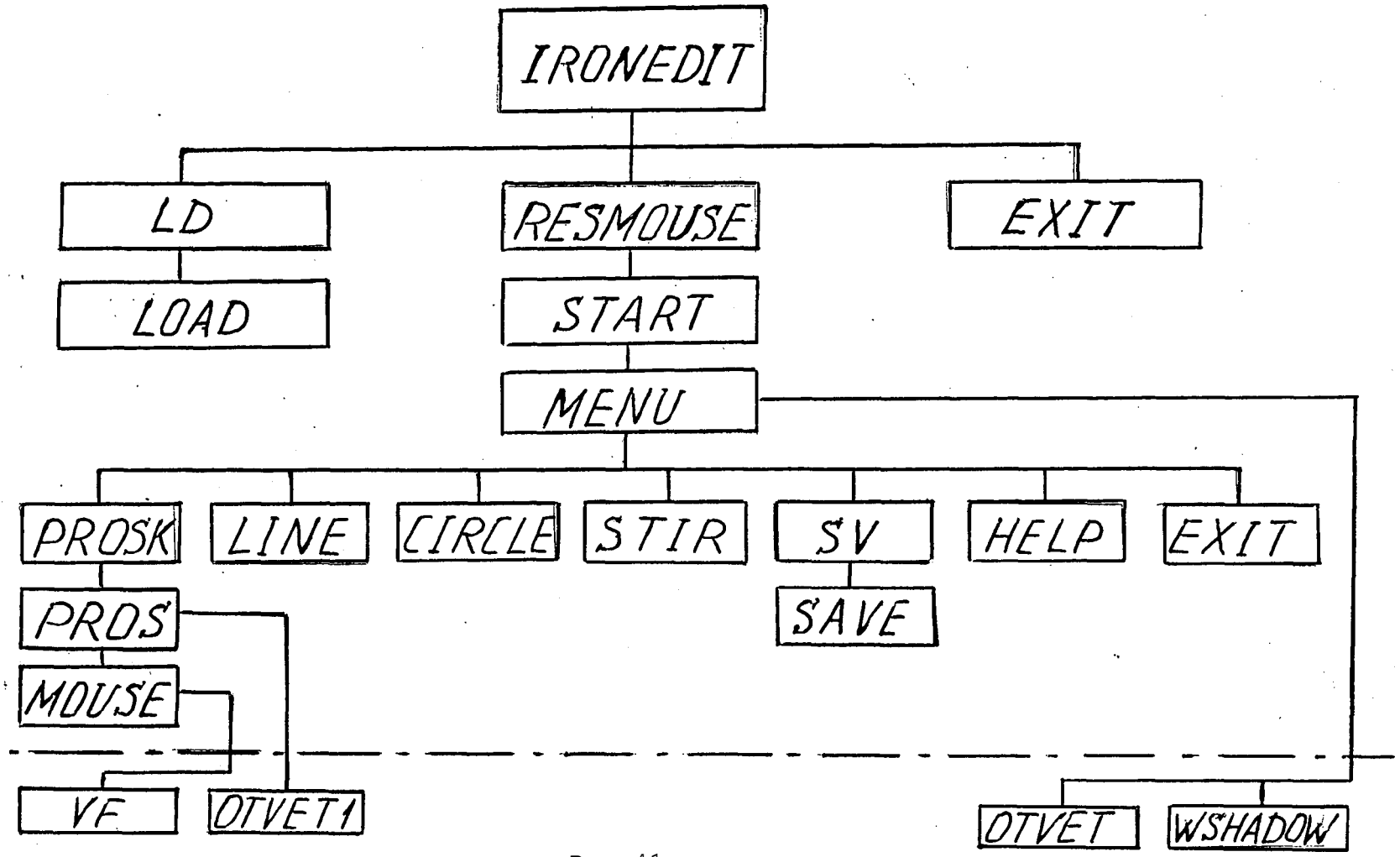


Рис.41

Модуль ''LOAD” осуществляет процедуру загрузки редактируемой станции. Вначале модуль передает управление модулю "RESM0USE", где осуществляется проверка наличия драйвера ''мыши". После возвращения из "RESM0USE" происходит установка графического режима и создание виртуального буфера. Затем, загрузив файл в виртуальный буфер, управление передается моулям "RAMA", где осуществляется рисование рамок и "MENU”. После этого (по возвращению из модуля "MENU") модуль по окончанию своей работы передает управление модулю "LD”.

Модуль "RESMOUSE" Сначала он осуществляет проверку наличия в памяти драйвера "мыши". Если драйвер установлен, то управление передается модулю "L0AD", либо модулю "IRONEDIT". Если драйвер не установлен, то выдается предупреждение о том, что "мышь" не инсталирована и прийдется выйти в DOS исправить погрешность. Далее модуль выходит в DOS.

Модуля "START” осуществляет установку режима графики для рисования на экране, затем происходит загрузка файла HELP на другую страницу экрана. После этого создается виртуальный буфер и далее управление передается модулю "MENU". По возвращении из "MENU” работа продолжается в модуле "IRONEDIT".

Модуль "MENU” выводит меню редактора-, на экран. После того, как на экране появится меню, пользователю предоставляется возможность выбрать одну из функций меню. В зависимости от выбранной функции вызовется соответствующий модуль. При выходе из меню управление передается либо модулю "L0AD”, либо модулю "START".

Модуль "PROSK" - основное меню просмотра выводит на экран подменю с переменными в нем базами данных. Пользова-

тель выбирает необходимую ему базу данных, далее проверяется наличие файла на диске с такой базой данных. Если этого файла не существует, то управление передается модулю "PRED", где выводится предупреждение. В случае существования такого файла база данных открывается и происходит вызов модуля "PROS".

Модуль "PROS" выводит на экран содержимое вызванной пользователем базы данных по 10 элементов. Если из выведенных названий нет нужного пользователю, то осуществляется возможность просматривать базу данных функциями "Ниже" и "Выше". При выборе функции "Ниже" производится установка указателя на 10 элементов вперед. Если конец базы, то на экране выводятся последние 10. элементов, если "нет", то можно работь дальше. При выборе функции "Выше" производится установка указателя на 10 элементов назад. Если конец базы, то на экране остаются последние 10 элементов. Если "нет", то можно далее осуществлять функцию "Выше". При выборе функции "Поиск" необходимо ввести название элемента. Далее, если элемент найден, то указатель устанавливается на названии элемента. Если элемент не найден, то выводится вопрос "Продолжать?". При ответе "Да". необходмо ввести название элемента и проделать вышеописанные операции. Если "Поиск" не нужен и необходим "Выход", модуль возвращается в "PROSK".

Процедуру рисования линии выполняет модуль "LINE". Сначала осуществляется очистка экрана и восстановление его из виртуального буфера. Далее осуществляется инициализация стартовой позиции для курсора "мыши”. Устанавливаем KL=0 и производим опрос состояния клавиатуры. Если ЕЮ не нажата, то

происходит опрос кнопок "мыши". Если нажата левая кнопка и KL=1, то фиксируется линия. Если KL 1, фиксируется точка, из которой будет начинаться прямая линия. Дальше опять устанавливаем KL=1 и можно рисовать линию одновременно с движением курсора (линия будет "тянуться" за курсором) "мыши". Если нажата правая кнопка, то осуществляется отмена рисования и KL=0, далее опять осуществляется опрос клавиатуры и состояния кнопок. Если нажата ЕЮ, то модуль возвращается в "MENU".

Модуль "CIRCLE" выполняет процедуру рисования окружности. Алгоритм модуля полностью аналогичен предшествующему. Отличие состоит в том, что вместо рисования линии изображается окружность. Сначала фиксируется центр окружности, а затем, удаляя или приближая курсор "мыши" к центру, можно получить нужную окружность. Затем модуль возвращает управление "MENU".

Модуль "STIR" осуществляет процедуру стирания изображенного на экране при помощи курсора "мыши". После очистки экрана и его восстановления из виртуального буфера происходит установка курсора "мыши" в начальную позицию. Затем производится опрос состояния-, клавиатуры. Если ЕЮ не нажата, то происходит опрос состояния кнопок "мыши". Если нажата левая кнопка, то происходит изменение вида курсора "мыши". Если левая кнопка удерживается в нажатом состоянии, то под курсором при его передвижении производится удаление. Если кнопка не удерживается в этом состоянии, то производится повторный контроль состояния левой кнопки. Затем производится проверка состояния правой кнопки. Если правая кнопка нажата, то вое-

станавливается вид. курсора-"мыши". Затем следует опрос состояния клавиатуры. Если ЕЮ нажата, то модуль передает управление модулю "MENU".

Процедура ввода имени сохраняемого файла осуществляется модулем "SV". После ввода с клавиатуры имени сохраняемого файла производится контроль случайного нажатия клавиши <ENTER>. Если имя не было введено, выдается сообщение о повторе действия. В противном случае модуль прекращает свою работу. При положительном ответе необходимо произвести вновь ввод имени файла. После ввода имени проверяется наличие файла с таким именем. Если файла не существует, выдается соответствующее сообщениие и передается управление модулю "SAVE". Если файл существует, то модуль запрашивает перепи-сать ли его. При утвердительном ответе начинает свою работу модуль "SAVE". При отрицательном - управление возвращается модулю "MENU".

Модуль "SV” для своей работы использует модуль "SAVE", который осуществляет процедуру записи файла на диск. После вывода топологии станции на экран из виртуального буфера осуществляется запись изображения на диск. Затем выдается сообщение об успешной записи файла и восстанавливается изображение на экране. Модуль передает управление модулю "MENU".

Модуль "СОМГНОЕ"производит все начальные установки. После включения графического режима производится загрузка файла в нулевую, затем в первую страницы. После опроса состояния клавиатуры при ненажатых ее клавишах происходит включение нулевой страницы экранаПри нажатии любой клавиши включается первая страница экрана. Затем происходит повторный опрос

состояния клавиатуры. При нажатой F10 включается текстовый режим и программа завершает свою работу.

4.13. Выводы по 4-ОИ главе

Микропроцессоры (МП) и микро-ЭВМ, благодаря реализованной в них возможности программного управления, обладают свойствами универсальных устройств и позволяют применять средства и методы цифровой обработки данных и цифрового управления в таких областях техники и народного хозяйства, в которых ранее их использование было экономически неоправданным.

Имеющийся опыт' применения МП и микро-ЭВМ в локальных устройствах и системах управления объектами и технологическими процессами свидетельствует о том, что использование МП обеспечивает достижение высоких технико-экономических показателей и расширение функциональных возможностей этих устройств и систем.

Не избежал напора МП и микро-ЭВМ и железнодорожный транспорт. В ряде стран (Япония, Англия, ФРГ) были созданы микропроцессорные системы управления стрелками и сигналами (микропроцессорная централизация - МПЦ). В нашей стране работы по МПЦ велись по двум организационным направлениям:

1. Совместно со странами: Болгария, Польша, Чехословакия, Германия, Румыния, Венгрия;

2. По заказу и под.общим руководством Главного Управления сигнализации и связи МПС.

С 1987 года начался новый виток исследования проблем,

связанных с созданием МПЦ, на этот раз под руководством института ГТСС. ХИИТ принял участие в разработке предложений по структуре технических средств МПЦ и методам определения их надежности: структуре программного и информационного обеспечения. Рекомендации по этим проблемам являются обобщением опыта ХИИТа по разработке микропроцессорной централизации, завершившейся созданием действующего макета.

Позднее работы были продолжены по заказу ''Укрзалізниці''.

МПЦ обладает рядом особенностей, стимулирующих поиск новых технологических принципов. К ним следует отнести:

принадлежность МПЦ к высоконадежным управляющим системам;

принадлежность ж.д. станции к классу гомогенных.

Ошибки в программном обеспечении МПЦ могут вести к катастрофическим последствиям. Это обстоятельство определяет максимальное использование всех методологических и технологических ресурсов независимо от затрат. В этом случае общую стратегию проектирования программного обеспечения МПЦ можно представить в следующем виде: задается минимально допустимый уровень надежности и тестирование и отладка ведутся до тех пор, пока этот уровень не будет достигнут. Очевидно, что стоимость ПО как изделия будет зависеть' от технологии его проектирования.

2 .Сделанный автором вывод о принадлежности железнодорожной станции к классу гомогенных дает возможность

применить к МПЦ предлагаемую в главе 2 стратегию проек-

тирования программного обеспечения;

реализовать такие технологические процессы, в которых большая часть технологических шагов - общая для всех станций.

3 .Произведенная оценка стоимости разработки технологии проектирования МПЦ, ориентированной на множество станций, позволила сделать вывод об экономической целесообразности единой для всех объектов управления ТП.

4 . Стратегия проектирования программного обеспечения микропроцессорной централизации дает возможность выяснить содержание принципов генерации ПО МПЦ, основанных на применении методов минимизации условно-постоянной составляющей модулей, сведении ее к постоянной составляющей. Среди них:

предусмотреть создание абсолютной библиотеки программных модулей, из Которой можно генерировать тексты для конкретной станции. Для этого использовать принципы функциональной избыточности и функциональной избирательности;

решение лингвистических задач описания объекта;

использование современных методов проектирования программного обеспечения на всех этапах разработки технологии проектирования, начиная с внешних спецификаций;

ввести обязательный контроль достоверности ПО по методу, предложенному в главе 3.

5 .Технологические методы проектирования и разработки программного обеспечения МПЦ, использовавшиеся на прошедшем этапе, могут быть оправданы только новизной проблемы. Такие аспекты технологии проектирования, как создание автоматизированных систем, не затрагивались вообще. Разработка велась

для конкретной станции и носила "поисковый", экспериментальный характер без учета возможности поставки системы в будущем на какую-либо другую станцию. Автор считает возможным автоматизировать процесс генерации текстов ПО МПЦ. Выделены основные из этапов автоматизации :

анализ всего разработанного в настоящее время ПО с разграничением системного и прикладного ПО;

выбор средств и автоматической генерации той части ПО, которая является уникальной, в том числе: выяснить какова должна быть лингвистическая поддержка объекта; создать формальную закодированную Информационную модель, по которой затем осуществляется привязка к конкретной станции; ввести автоматический контроль вводимой информации; предусмотреть получение цифровой модели станции с последующим выводом на экран топологии станции для непосредственной сверки информационной модели с документом-планом станции, что значительно сократит количество ошибок при описании объекта; в конечном счете добиться автоматического формирования массивов статической модели станции;использовать появляющуюся специфическую информацию для подстройки на конкретный объект.

В перспективе предполагается автоматизация наиболее рутинных процессов в проектировании и, в первую очередь, таких как .создание наборов данных, отражающих особенности конкретной станции, что позволяет повысить такие конструктивные особенности ПО как корректность на стадии разработки и надежность на стадии эксплуатации.

6 . В работе не ставится задачи автоматизации МПЦ в целом (это под силу только большому коллективу разработчиков), а

на одной из подсистем ПО МПЦ продемонстрированы технологические методы проектирования ПО СУ гомогенными объектами.

Понятие гомогенности как абстрактной характеристику ОУ практически отображается в ПО СУ подобными объектами, поэтому концепция гомогенности применима и к нему. Понятие гомогенности применяется к программному обеспечению систем управления объектами, относящимися к классу гомогенных, ***и к*** отдельным его частям, образованным при декомпозиции ПО на составляющие. Так к типу гомогенных можно отнести и диалоговую подсистему МПЦ. В ней алгоритмы монитора системных сообщений, синтаксического контроля директив, коды об ошибках, массивы кодов директив и их числовых эквивалентов не зависят от конкретной станции. То есть можно предположить, что эти алгоритмы будут разрабатываться один раз.

Однако, статическая модель станции, отображаемая на цветном графическом терминале, технологические номера стрелок, рельсовых цепей, сигналов и их количество (а, следовательно, ***и*** размеры соответствующих массивов) для каждой конкретной станции различны. Так как алгоритмы - общие, то эта часть может быть разработана один раз. А информация, имеющая уникальный характер, может генерироваться для каждой станции, то есть она и является объектом автоматизации.

7 . Ввиду того, что информационная составляющая является наиболее насыщенной и подверженной изменениям, автор остановился на системе проектирования подсистемы отображения информации - пакете программ проектировщика описания станции.

ЗАКЛЮЧЕНИЕ

В результате рассмотренных в работе вопросов можно сделать следующие выводы. .

1. В настоящее время проблема качества программного обеспечения выражается в сопоставлении двух противоречивых и тесно связанных его показателей: стоимости и надежности. Решение этой проблемы в пределах ''стоимость-надежность'' перспективы не имеет. Снижение стоимости изделия за счет надежности - путь тупиковый, и, в то же время, достижение требуемого уровня надежности любой ценой часто оказывается неприемлемым - предел допустимых затрат на изготовление достигается раньше, чем требуемый уровень надежности. Решение этой проблемы следует искать в некоторой метасистеме, для которой показатели ''стоимость” и "надежность” будут иметь минимальную зависимость друг с другом. Такой системой применительно к программному обеспечению является процесс его создания - технология его проектирования.

2. В ходе исследований выявлено, что эффективность технологии проектирования, конкретное количество ее этапов и их продолжительность будет зависеть от:

выбранного критерия эффективности технологии проектирования (максимальныйуровень надежности при снижении стоимости технологических шагов),

характера разрабатывемой системы (для СУ характерным является значительное сокращение этапа сопровождения и повышение роли всех предыдущих этапов),

использования современных методов и средств реализации

всех технологических шагов, начиная с внешних спецификаций, позволяющих достигать максимального качества программ при снижении их стоимости,

особенностей объекта управления, для которого разрабатывается система.

3. Поставлена основная цель исследования: поиск методики разработки программного обеспечения для функционально однородных объектов управления, позволяющей создавать надежное эффективное программное обеспечение при снижении затрат на его разрааботку.

4. Введена математическая модель всего класса функционально однородных объектов, названных гомогенными (от греческого ''homogenes'' - однородный). Выделение гомогенных объектов в отдельный класс дало возможность:

предложить стратегию проектирования программного обеспечения в целом,

реализовать такие технологические процессы, в которых большая часть технологических шагов - общая для выделенных объектов,

при проектировании программного обеспечения для конкретных объектов управления сконцентрировать усилия только на тех его составляющих, которые являются носителями различий между объектами данного класса.

5. Предложенная стратегия проектирования программного обеспечения дала возможность выяснить содержание принципов генерации ПО для систем управления гомогенными объектами. Среди них выделены:

принципы функциональной избирательности и функциональной

избыточности,

лингвистические задачи генерации,

контроль достоверности ПО.

6. Описанная методика проектирования программного обеспечения могут быть использованы при разработке микропроцессорной системы управления стрелками и сигналами на ж.д. станциях. Технологические принципы проектирования программного обеспечения МПЦ, использовавшиеся на прошедшем этапе, могут быть оправданы только новизной проблемы. Разработка велась для конкретной станции и носила "поисковый", экспериментальный характер, без учета возможности поставки системы в будущем на какую-либо другую станцию.

К особенностям МПЦ, стимулирующим поиск новых технологических прнинципов, следует отнести :

принадлежность МПЦ к высоконадежным управляющим системам,

принадлежность ж.д. станции к классу гомогенных.

7. Исходя из принадлежности ж.д. станции к классу гомогенных сделан вывод о :

применении к МПЦ предлагаемой стратегии проектирования программного обеспечения систем управления гомогенными объектами,

реализации таких технологических процессов, в которых большая часть технологических шагов - общая для всех станций,

использовании принципов генерации программного обеспечения для систем управления гомогенными объектами,

возможности автоматизации процесса генерации ПО МПЦ.

8. На примере системы проектирования подсистемы отображения информации проиллюстрированы технологические методы проектирования ПО СУ гомогенными объектами (разработан пакет проектировщика описания станции).

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Акита К. Разработка системы микропроцессорной централизации SMILE // Железные дороги мира.-1987.-Jfc 8.- С.42-44.

2. Арефьева Н.А., .Пушкина И.П., Родионов С.Т. H1P0-технология - метод разработки и сквозного документирования программ по принципу "сверху вниз”// Управляющие системы и машины.-1978.-J6 3.- С.35-39.

3. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов.-М.: Мир, 1979.

4. Баррон Д. Введение в языки программирования.-М.: Мир, 1980.- 190 с.

5. Безопасность программного обеспечения систем СЦБ на базе ЭВМ // Железные дороги мира.-1987.-J6 10.- С.75-76.

6. Большая Советская энциклопедия /Гл.ред. А.М.Прохоров. 3-є изд.-М.:Советская энциклопедия, 1972.-т.7.- С.53.

7. Боэм Б., Браун Дж. и др. Характеристики качества программного обеспечения.-М.: Мир, 1981.- 206 с.

8. Борковский А.Б. и др. Словарь по программированию (англ., русск., нем., франц.)-М.:Русский язык, 1991.- 286 с.

9. Братчиков И.Л. Синтаксис языков программирования.-М.:Наука, 1975.- 232 с.

10. Будя А.П., Кононюк А.Е. и др. Справочник по САПР. -К.: Техніка, 1988.- 375 с.

11. Ван дер Варден Б.А. Алтебра.-М:Наука,.1979.- 240 с.

12. Василенко М.Н., Бакалов С.П., Румин М.Я. Автоматизация проектирования программного обеспечения

//Микропроцессорные системы на железнодорожном транспорте: Сб. науч, тр./ ЛИИЖТ.-Ленинград, 1991.- С.12-15.

13. Вельбицкий И.В. Технология программирования.-Киев: Техніка, 1984.- 280 с.

14. Вирт Н. Систематическое программирование: Введение.- М.: Мир, 1977.- 181 с.

15. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ: Пер. с англ. - М.: Мир, 1985.- 332 с.

16. Выбор и обоснование вариантов построения электрической централизации ЭЦ-Е: Отчет о НИР.-ГТСС.-ГР 418742003.1А.- Ленинград, 1987.-ИСПОЛН.: Пресняк С.С., Добрянский В.М., Зорина Е.И. и др.

17. Гантер Р. Методы управления проектированием программного обеспечения.-М.: Мир, 1981.-390 с.

18. Гарелли Ф. Система микропроцессорной централизации железных дорог Франции // Железные дороги мира.-1990.4.- С.64-65.

19. Гладков В.А. Синтез аппаратных структур автоматизированной системы управления технологическим процессом по критериям надежности и безопасности : Автореф. дис....канд.-техн.наук. - Киев, 1989.

20. Глушков В.М. Фундаментальные исследования и технология программирования //Программирование.-1980.-J6 2.- С.3-13.

21. Глушков В.М., Вельбицкий И.В. Технология программирования и проблемы ее автоматизации //Управляющие системы и машины.-1976 .-№ 6.- G.75-93.

22. Глушков В.М., Капитонова Ю.В., Летичевский А.А.

Автоматизация проектирования вычислительных машин.-К.: Наук, думка, 1975.- 231 с.

23. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки программирования.-К.: Наук, думка, 1978.- 320 с.

24. Горский Д.П. и др. Краткий словарь по логике.-М.: Просвещение, 1991.- 208 с.

25. Глазунов Л.П. ,Грабовецкий В.П., Щербаков О.В. Основы теории надежности автоматических систем управления. -Ленинград: Энергоатомиздат, 1984.- 208 с.

26. Гласс Р. Руководство по надежному программированию.- М.: Финансы и статистика, 1982.- 256 с.

27. Глушков В.М., Вельбицкий И.В. Технология программирования и проблемы ее автоматизации // Управляющие системы и машины.-М, 1976.- N 6.- 0.75-93.

28. Гнеденко В.В. и др. Математические методы в теории надежности.-М.: Наука, 1965.- 542 с.

29. Голинкевич Р.А. Прикладная теория надежности.-М.: Высшая школа, 1985.- 168 с.

30. Гопанов П.Г., Липаев В.В., Волков В.А., Просви-' рин В.Н. Анализ ошибок в сложных программах систем управления объектами //Управляющие системы и машины. 1973.- N 3.- 0.31-38.

31. Гусейлова А.А., Никитин А.М. Вычислительные системы повышенной отказоустойчивости //Управляющие системы и машины.-1987.-J6 5.- 0.25-30.

32. Гюнтер И. Проверка программного обеспечения систем микропроцессорной централизации //Железные дороги мира.- 1990.- J6 8.- 0.41-44.

33. Дейкстра Э. Дисциплина программирования.-М.: Мир, 1978.- 274 с.

34. Добрянский В.М., Донина С.Л., Кришталь Н.П. Синтез рабочей программы управляющей цифровой ЭВМ из функционально избыточного набора модулей // Автоматизированные системы управления и приборы автоматики.-Харьков: ХГУ, ''Вища школа", 1985.- С.81-86.

35. Добрянский В.М. Принципы построения программного обеспечения микропроцессорных систем, управляющих технологическими процессами // Автоматизированные системы управления технологическими процессами на ж.д. станциях: Межвуз. сб. науч, тр./ ДЛИТ.-Днепропетровск, 1983.- С.81-85.■

36. Добрянский В.М., Зорина Е.И. Принципы унификации прикладного программирования для микропроцессорной централизации // Микропроцессорные системы управления и устройства контроля на ж.д. транспорте :Межвуз. сб. научи, тр. / ХИИТ. - Харьков, 1986.- С.88-91.

37. Добрянский В.М., Зорина Е.И., Майдан Б.В. Алгоритмический способ контроля и восстановления вычислительного процесса при сбоях аппаратуры // Применение микропроцессорных устройств в системах железнодорожной автоматики: Сб. научн. тр. / ХИИТ.- Харьков, 1988.- Вып.7.- С.64-66.

38. Добрянский В.М., Зорина Е.И., Казанко В.В. Автоматизированное распределение функций между элементами управляющей микропроцессорной сети // Микропроцессорные системы связи и управления на ж.д. транспорте: Тезисы докл. республ. ксифер. - Алушта-Киев, 199(3.- G.3-5.

39. Добрянский В.М., Зорина Е.И. Принципы

избыточности и функциональной избирательности при проектировании программного обеспечения МПЦ // Микропроцессорные системы связи и управления на ж.д. транспорте : Тезисы докл. республ. школы-семинара.-Алушта-Киев, 1993.- 0.36.

40. Добрянский В.М., Зорина Е.И., Браило Ф.В. Автоматизированная система прикладного программирования для систем, управляющих гомогенными объектами // Микропроцессорные системы связи и управления на ж.д. транспорте: Тезисы докл. республ. школы-семинара.-Алушта-Киев, 1993.- 0.40.

41. Добрянский В.М., Зорина Е.И. О математической модели некоторого класса объектов //Элементы и устройства современных систем железнодорожной автоматики: Сб.научн.тр. /ХИИТ.-Харьков, 1993.-ВЫП.23.- 0.29-32.

42. Добрянский В.М., Майдан Б.В. Программные методы повышения надежности системы микропроцессорной централизации //Проблемы повышения надежности и безопасности техн.средств на ж.д.транспорте: Тез.докл. Всес.научно-практ. конф, с уч. специалистов соц.стран: М., 7-9 июня 1998.- 0.137-139.

43. Довгаль С.И., Литвинов Б.Ю., Сботнев А.И. Персональные ЭВМ: Турбо Паскаль V 6.0. Объектное программирование. Локальные сети.-Киев: Информсистема сервис, 1993.- 440 с.

44. Дополнительный НИР по разработке и обоснованию вариантов построения системы микропроцессорной централизации: отчет о НИР/ ХИИТ.-ГР 01870044080.-Харьков, 1987.-.180 с. - исполн. Загарий Г.И., Добрянский В.М., Зорина Е.И. и др.

45. Дыкина Н.В. Безопасность и надежность систем программного обеспечения с избыточностью // Вестник ВНИИЖТ,- 1991.- J6 2.- С.44-47.

46. Диллон Б., Сингх И. Инженерные методы обеспечения надежности систем:-М.:Мир, 1984.- 318 с.

47. Дружинин Г.В. Надежность автоматизированных систем.- М.: Энергия, 1977.- 536.с.

48. Дружинин Г.В. Методы оценки и прогнозирования качества.-М.: Радио и связь, 1982.- 160 с.

49. Ершов А.П. Введение в теоретическое программирование.-М.: Наука,'1977.- 288. с.

50. Ершов П.П. Опыт интегрального подхода к актуальной проблематике программного обеспечения // Кибернетика.-1984.- N 3.- С.11-21.

51. Ефимов В.Ю., Денисов Н.П. Микропроцессорная централизация стрелок и сигналов //Применение соврем, техн, средств автомат, и вычисл. техники в системах упр. перевозочным процессом.-Л.,1988.- С.3-8.

52. Зиглер К. Методы проектирования программных систем.- М.: Мир, 1985.

53. Зуве К.-Х. Электронная централизация фирмы SIEMENS //Железные дороги мира.-1984.-J® И.- С.23-27.

54. Зуве К.-Х. Централизация стрелок и сигналов на базе ЭВМ //Железные дороги мира.-1984.-J6 2.- С. 15-20.

55. йодан Э. Структурное проектирование и конструирование программ.—М.: Мир, 1979.- 415 с.

56. Иосимура Х и др. Внедрение микроэлектроники в устройствах сигнализации //Железные дороги мира.-1985.-16 8.- С.6-9.

57. Инду К.А., Касаткин А.И., Бахтизин В.В. Прогнозирование надежности программ на ранних этапах разработки //

Надежность и контроль качества.-1982.-}& 5.- С.1-10.

58. Каазик Ю.А. Математический словарь.-Таллин: Вангус, 1985.- 284 с.

59. Кавалерчик Б.Я. Надежность программного обеспечения и условия эксплуатации // Микропроцессорные средства и системы.-1987.-N 3.- С.20-23.

60. Казаков А.А. Электрическая централизация стрелок и сигналов.-М.:Транспорт, 1970.

61. Катков А.В., Сепетый А.А. Вопросы программной диагностики в управляющих микропроцессорных системах /7Тр. Ростов.ин-та инж.ж.д.тр-та, 1987.-№ 188.- 0.27-38.

62. Керимов З.Г., Багиров С.А. Автоматизированное проектирование конструкций.-М.: Машиностроение, 1985.- 222 с.

63. Клини 0., Весли Р. Основания интуиционистской математики. -М.: Наука, 1978.

64. Клини С. Математическая логика.-М.: Мир, 1973.

65. Кобринский Я.Н. Оценка разрешающей способности оперативных методов контроля сохранения информации в постоянной памяти.// Автоматика и телемеханика. -1982.-J6-4.- 0.105-116.

66. Коваленко А.Е., Гула В.В. Отказоустойчивые микропроцессорные системы.-Киев: Техніка, 1986.- 150 с.

67. Кокурин М.И. Эксплуатационные основы железнодорожной автоматики и телемеханики.-М.: Транспорт, 1980.- 165 с.

68. Королев Л.Н. Структуры. ЭВМ и их математическое обеспечение.-М.: Наука, 1974.- 254 с.

69. Котляренко Н.Ф., Гладков В.А. и др. Опыт разработки микропроцессорной централизации стрелок и сигналов в СССР //Сборник докладов научн.техн.конф. (г.Бухарест, 1985). М.:

Транспорт, 1987.- 0.287-295.

70. Кузин Л.Т., Стрижевский В.С. Проблемы создания и внедрения современных технологий программирования // Прогрессивные технологии программирования.-М.: Моск, дом науч.-техн. пропаганды, 1983.- 0.3-7.

71. Кулаков А.Ф. Оценка качества программ ЭВМ.-Киев: Техніка, 1984.- 168 с.

72. Куплер М.Н. Проектирование систем микропроцессорной централизации //Железные дороги мира.-1987.-J® 2.- 0.43-46.

73. Кушниренко А.Г.,Лебедев Г.В. Программирование для математиков.- М.:Наука; Гл.ред.физ-мат.лит., 1988.- 384 с.

74. Кушниров В.А. Некоторые аспекты технологии программирования // Управляющие системы, и машины.-М., 1975.-N 6. - 0.12-16.

75. Малхасян М.Н. Некоторые способы создания отказоустойчивых микропроцессорных вычислительных систем //Вычислительные ср-ва инф.упр.систем, М., 1988.- 0.88-94.

76. Ларионов А.М., Майоров С.Л. и др. Вычислительные комплексы, системы и сети.- Л.:Энергоатомиздат, 1987.- 287с.

77. Липаев В.В., Потапов А.И. Длительность разработки• сложных программных средств // Микропроцессорные средства и системы.-1986.-N 6.- 0.17-20.

78. Липаев В.В. Тестирование программ.-М.: Радио и связь, 1986.- 296 с.

79. Липаев В.В. Качество программного обеспечения.-М.: Финансы и статистика, 1983.- 263 с.

80. Липаев В.В., Гбпанов П.Г., Просвирин В.Н. Оперативный контроль функционирования алгоритмов управляющих систем

в реальном масштабе времени // Управляющие системы и машины. -1973.-N 2,- С.62-68.

81. Липаев В.В., Филиппович В.В. Принципы и правила модульного построения сложных комплексов программ АСУ // Управляющие системы и машины.-1975.-N 1.- С.15-22.

82. Левин В.И. Логическая теория надежности сложных систем.-М.: Энергоатомиздат, 1985.- 128 с.

83. Левицкий Н.К., Маничев М.Н. Диспетчерское руководство на промышленном транспорте.-М.: Транспорт, 1976.- 184 с.

84. Литвинов В.А., Крамаренко В.В; Контроль достоверности и восстановление информации в человеко-машинных системах. -К.: Техніка, 1986.- 200 с.

85. Лонгботтом Р. Надежность вычислительных систем. -М.: Энергоатомиздат, 1985.- 288 с.

86. Лукин Н.Н. Надежность программного обеспечения и ее влияние на достоверность обработки информации в АСУТП // Автоматизация процессов технического обслуживания систем интервального регулирования движения поездов: Межвуз. темат. сб.науч. тр./ Омский ин-т инж. ж.д.тр-та.-Омск, 1984.- 124 с.

87. Майерс Г. Искусство тестирования програми.-М.: Финансы и статистика, 1982.- 176 с.

88. Майерс Г. Надежность программного обеспечения. -М.: Мир, 1980.- 360 с.

89. Мясников В.А. Совершенствование технологии программирования - важнейшая народнохозяйственная задача // Управляющие системы и машины.-1980.-Jfc 1.- С.6-8.

90. Надежность и эффективность АСУ /Под ред. Ю.Г.Зарецина.-Киев:Техника, 1975.- 368 с.

1.91 Николаев Р.А. и др. Проблемы повышения достоверности в информационных системах.-Л.: Энергоатомиздат. Ленингр. отд-ние, 1982.- 138 с.

92. Огнев И.В., Сарычев К.Ф. Надежность запоминающих устройств.-М.: Радио и связь, 1988.- 244 с.

93. Организация процесса проектирования в безбумажной Р-технологии программирования /А.А.Тараненко, В.Ю.Каюров, В.Ф.Кирсанов и др. //Управляющие системы и машины.-1982.- J£ б.- С.38-43.

94. Оре О. Теория графов.-М.: Наука, 1980,- 336 с.

95. Панич Н. Концепция создания микропроцессорной централизации //Железопыт.трансп.-1968.-^ 1.- С. 16-21.

96. Парасюк И.Н., Сергиенко И.В. Модульный подход к построению семейства пакетов прикладных программ. //Программирование. -1981 6.- С.29-34.

97. Першиков В.И., Савинков В.М. Толковый словарь по информатике.-М.: Финансы и статистика, 1991.- 543 с.

98. Петренко А.И., Семенков О.И. Основы построения систем автоматизированного проектирования.-К.: Вища школа, 1965.- 294 с.

99. Пивоваров А.Н. Методы обеспечения достоверности информации в АСУ: Обзор методов и факт, данные.-М.: Радио и связь, 1982.- 144 с.

100. Половко А.М. Основы теории надежности.-М.:Наука. -1964.

101. Применение микропроцессорных систем централизации и блокировки на Британских ж.д. //Железные дороги мира.-1987. -Ji 8.- С.72-74.

102. Разработка системы команд и программного обеспечения для модернизированных устройств логической обработки информации:Отчет о НИР /ХИИТ.-ГР 01840017693.- Харьков,1984.- 129 с.-Исполн.: Кедрус В.А., Добрянский В.М., Зорина Е.И., Калиновский А.В. и др.

103. Разработка программного обеспечения для микропроцессорных устройств "Старт" и "Сервис":Отчет и НИР /ХИИТ.-ГР 01860026051.-Харьков, 1986.- 178 с.- Исполн.: Кедрус В.А., Добрянский В.И., Зорина Е.И., Махота А.Ф. и др.

104. Родионов С.Т., Попов П.Г. Использование Р-технологии для построения информационно-поисковых систем //Управляющие системы и машины.- 1980.-Jfc 1.- С.94-98.

105. Рябинин И.А., Черкесов Г.Н. Логико-вероятностные методы исследования надежности структурно-сложных систем.- М.: Радио и связь, 1981.- 264с.

106. Сакман Г. Решение задач в системе человек-ЭВМ.- М.: Мир, 1973.

107. Сапожников В.В. Об одной структуре микропроцессорной централизации //Применение совр.техн.средств автомат, и выч.техники в системах упр. переводочным процессом.-Л.- 1988.- С.68-73.

108. Системы автоматизированного проектирования./ Под ред. Дж.Аллана.-М.: Наука; Гл. ред. физ-мат.лит., 1985.- 376 с.

109. Системы микропроцессорной централизации //Железные дороги мира.-1987.-№ 4.- С.76-77.

110. , Словарь иностранных слов /Под ред. Ф.Н.Петровой, 11-е изд.-М.: Русский язык, 1984.- 608 с.

111. Тамм Б.Г., Тыугу Э.Х. О создании проблемно-ориентированного программного обеспечения //Кибернетика.-1975.-Я 4.- С.76-85.

112. Тищенко Н.М.. Введение в проектирование систем управления.-М.: Энергоатомиздат, 1986.- 248 с.

ИЗ. Турский В. Методология программирования.-М.: Мир, 1981.- 263 с.

114. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ.-М.: Мир, 1981.- 320 с.

115. Фуксман А. Л. Технологические аспекты создания-программных систем.-М.: Статистика, 1979.- 184 с.

116. Функциональная структура 0С/360/ -М.: Советское радио, 1971.

117. Хьюз Дж., Мичтом Дж. Структурный подход к программированию.-М.: Мир, 1980,- 278 с.

118. Цветков А.А. Технология разработки пакетов прикладных программ как промышленных изделий //Технология программирования: Тез. докл. I Всесоюз. конф. Секция 2.-К.: Ин-т кибернетики АН УССР, 1979.- С.59-60.

119. Цилковски Л. Разработка микропроцессорной системы СЦБ для станций польских железных дорог //Автоматика и вычислительная техника на ж.д.транспорте, Л.-1986.- С.106-109.

120. Централизация стрелок и сигналов на микропроцессорах (МПЦ). Разработка технического задания на трехканальную МПЦ. Часть 3. Программное обеспечение, имитационное моделирование и принципы диагностирования МПЦ: Отчете НИР/ХИИТ.-ГР 01850014773-Харьков, 1985.- 118 с.-Исполн.:Кедрус В.А., Добрянский В.М., Зорина Е.И. и др.

121. Хенли Э.Дж.., Кермамото Х. Надежность технических систем и оценка риска.-М.:Машиностроение, 1984.-528 с.

122. Холстед М.Х. Начала науки о программах.-М.: Финансы и статистика, 1981.- 128 с.

123. Черноусов Е.А. Программирование задач обработки экономической информации.-М.: Финансы и статистика, 1982. - 160 с.

124. Шиханович Ю.А. Введение в современную математику. М.: Наука.-1985.- 376 с.

125. Шрейдер Ю.А. Равенство. Сходство. Порядок.-М.: Наука, 1971.- 256 с.

126. Шураков В.В. Надежность программного обеспечения систем обработки данных.-М.: Статистика, 1981.- 216 с.

127. Эфрос ЛТ.В. Концептуальный анализ программных систем // Управляющие системы и машины.-1979.-N 2.- С.25-32.

128. Якубовский С.В., Баранов Н.А., Ниссельсон Л.И., Ронешкин М.Н., Ушибышев В.А. Аналоговые и цифровые интегральные микросхемы.-М.: Радио И связь, 1984.- 432 с.

129. Ясухара Х. Разработка систем централизации на базе ЭВМ //Железные дороги мира.-1985.5,- С.28-32.

ПР ИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

ТЕКСТЫ ОСНОВНЫХ ПРОГРАММ ПРОЕКТИРОВАНИЯ ОПИСАНИЯ ЖЕЛЕЗНОДОРОЖНЫХ СТАНЦИИ

ТЕКСТ ПРОГРАММЫ ФОРМИРОВАНИЯ И ОБСЛУЖИВАНИЯ БИБЛИОТЕКИ ЭЛЕМЕНТОВ ЖЕЛЕЗНОДОРОЖНЫХ СТАНЦИИ

```
жжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжжо1

ЖЖ          ОСНОВНАЯ ПРОГРАММА БИБЛИОТЕКИ ЖЕЛЕЗНОДОРОЖНЫХ ЭЛЕМЕНТОВ             ЖЖЖЖ
жж                                                                              ЖЖЖЖ
iЖ 1Ж )Ж зЖ Ж )Ж )Ж1Ж<Ж з(( * зЖ )Ж Ж Ж )Ж jjc j(c jjc jjc jjc зЖ)((jjc j(c зЖ зЖ зЖ зЖ Ж
зЖ зЖ зЖ ж зЖ зЖ зЖ ЗЖ зЖ зЖ зЖ ЗЖ1ЖзЖ зЖ зЖ )Ж зЖ зЖ )Ж зЖ зЖ зЖ зЖ зЖ зЖ зЖ зЖ ЗЖ зЖ ’Ж
зЖ зЖ зЖ зЖ ’Ж зЖ зЖ зЖ зЖ
00,0 clear
set talk off
set echo off
set status off
set bell off
set scoreboard off
set console on
set cursor off
set escape off
set wrap on
set procedure to pcxlib
SET PROC TO _wshadow
SET PROC TO _wshrink
set procedure to _USERFUN
set procedure to —password

PUBLIC WnExpand, WnShrink,WnShadow,WnBorder,WnBackgr,WnSound

WnExpand=.t.                     && Раскрытие окон
WnShrink=.t.                     && закрытие окон
WnShadow=.. t.                   && Тени у окон
WnBorder= "                      && Рамки окон
WnBackgr=" y'                    && символ-заполнитель
WnSound=.T.                      && Сигналы при раскрытии/свертке



if .not.file('DGEDEFS.PRG')
  .do . ADGEDEFS
else
   do DGEDEFS
end if

do pcxlnit                             && инициализация PCX


set console off
pcxtype = pcxEGA_10
retcode = pcxSD(pcxtype)
retcode = pcxSM(pcxGRAPHICS)
pcxSP(l)

pict = "mat.pcx"

pal = SPACE(48)
retcode = pcxGFP(pcxtype,pict,pal)
retcode = pcxSDP(pal)
retcode = pcxFD(pict,0,0,0)

011,31 say '
012,33 say 'Вы вошли в программу'
```

```
 set color to r+/n
 613,37 say date()
 614,37 say time()

 set color to gr+/n,x
 parol = "relsa"
 password = space(5)
 615,31 say '
 616,31 say '
 617,31 say '    ПАРОЛЬ
 618,31 say '
 619,31 say '
 pcxSP(O)
 617,40 get password VALID password(password,parol)
 read
 set color to


»**pcxtype = pcxVGA_12
pcxSP(O)
pcxSM(pcxTEXT)

set cursor off
DO zvuk
60,0 CLEAR
SET COLOR TO W/N+

60,0,23,79 box
60,15 TO 2,62 DOUBLE

set color to gr+/b+
624,15 say chr(27)+chr(24)+chr(25)+chr(26)
624,col()+i say ’’-Перемещение ПО меню"
624,col()+5 say "ENTER - Выбор"

SET COLOR TO G+/N+
61,16 SAY ' БИБЛИОТЕКА ЭЛЕМЕНТОВ ЖЕЛЕЗНОДОРОЖНЫХ СТi=НЦИИ '
DO WHILE.T.
          SET COLOR TO W/N+,GR+/RB
          65,7,7,33 BOX CHR(176)
          64,6 TO 6,32 DOUBLE

          65,41,7,54 BOX CHR(176)
          64,40 TO 6,53 DOUBLE

          65,62,7,72 BOX CHR(176)
          64,61 TO 6,71 DOUBLE

          SET COLOR TO R/BG,GR+/RB
          65,7 PROMPT " ОБСЛУЖИВАНИЕ БИБЛИОТЕКИ "
          65,41 PROMPT " СЕРВИС
          65,62 PROMPT " ВЫХОД

          MENU to level1
```

```
DO CASE
CASE levell=l
DO OBLIB
CASE levell=2
DO servis
CASE level1=3
msg 1='ВЫ УВЕРЕНЫ,ЧТО ХОТИТЕ ВЬМТИ ?'
SET COLOR TO W/N+
@14,23,18,57 BOX CHR(176)
DO otvet
IF otvet=2
Z=1
DO WHILE Z<19
@13,21+Z,18,21+Z BOX CHR(178)
@13,58-Z,18,58-Z BOX CHR(178)
Z=Z+1
ENDDO
DO zvuk

loop
ENDIF

Z=1
DO WHILE Z<19
Q13,21+Z,18,21+Z BOX CHR(178)
@13,58-Z,18,58-Z BOX CHR(178)
Z=Z+1
ENDDO
DO zvuk
CLEAR
set color to w+/b+
@9,20 clear to 15,60
@9,20 to 15,60 double
@10,22 say '                                          *
@11,22 say '                                          *
@12,22 say '               конец работы              *
@13,22 say '                                          *
@14,22 SAY-■                                          *
clear all
close all
SET TALK ON
SET BELL ON
SET STATUS ON
CANCEL
ENDCASE
ENDDO
 clear
quit
```

ijc)(()jc sjc #)(( ) ((J (C)|(з(c)((ijc

Ф Ф W Ф Mf

»*******<<<<<<<<< функция, проверяющая правильность пароля >»>»»»*******»**

Ф U/ Ф Ф ф tit ф Ф Ф Ф Ф Ф & ф ф \1г 4

™ Л' ™ ® ® * Ж ® * Ж ® ® Ж ® Ф Ж * ф * п* * * * * * п» п* * ™ ч» * * р т» п* *

```
PARAMETERS getvar,word
IF getvar Oword
617,40 say ' НЕВЕРНЫЙ'
inkey(O)
#tD0 pict
close all
quit

endif
return .t.      && Если пароль верный
Ж**Ж***Ж#Ж*Ж#***********<< END OF PASSWORD >>*ЖЖЖЖ**Ж****ЖЖ**ЖЖЖ**ЖЖЖ**ЖЖЖ*ЖЖ


     Ф Ф          Ф                          Ф                                    4
    ф> zp ПХ ф ф              ф                                ф Ж *П * *          Ф       ™       * * * П» * Ф Ф Ф              Ф
Ж*                      PROCEDURE ОБСЛУЖИВАНИЕ БИБЛИОТЕКИ                                          **
    ф      Ф             Ф      U/ &                                        4


save screen to sc
level2=7
DO WHILE.NOT.level2=0
SET COLOR TO W/N+
011,7,19,33 BOX CHR(176)
010,6 TO 18,32 DOUBLE
SET COLOR TO R/BG
 011,7  PROMPT '        ПРОСМОТР
 012,7  PROMPT '        РЕДАКТИРОВАНИЕ
 013,7  PROMPT '        УДАЛЕНИЕ
 014,7  PROMPT '        ДОПОЛНЕНИЕ
 015,7  PROMPT '        ПОИСК
 016,7  PROMPT '        СПРАВКА
 017,7  PROMPT '        ВЫХОД
MENU TO level2
.DO CASE
CASE level2=l
DO prosk
restore screen from sc
CASE level2=2
DO red
restore screen from sc
CASE level2=3
DO del
restore screen from sc
CASE level2=4
DO dop
restore screen from sc
CASE level2=5
DO poisk
restore screen from sc
CASE level2=6
DO spravka
restore screen from sc
CASE level2=7
SET COLOR TO W/N+,GR+/RB
wshrink (10,6,18,32,WnBorder,sc)
```

```
close all                                                                    '
RETURN
ENDCASE
enddo
RESTORE SCREEN FROM sc
CLOSE ALL
RETURN
*******************************<< END of oblib >>******ЖЖ**********ЖЖ**ЖЖЖЖ


**                     ОСНОВНОЕ МЕНЮ ПРОСМОТРА                        ******



leve!3=6
DO WHILE.NOT.leve!3=0
SET COLOR TO GR+/B+
013,15 TO 19,41 DOUBLE
wshadow (13,15,19,41)
SET COLOR TO G+/B+
014,16 PROMPT     '    СТРЕЖИ
015,16 PROMPT     '    СВЕТОФОРЫ
016,16 PROMPT     ■    ШРИФТ
017,16 PROMPT     ‘    ПРОЧЕЕ
018,16 PROMPT     '    ВЫХОД
MENU TO leve!3
DO CASE
CASE level3=1
kl = 15

if .not.file('vecstrel.dbf')
im = "vecstrel.dbf"
DO pred with im
loop
endif

use vecstrel
         DO pros with kl
CASE level3=2
kl = 20

if .not.file('vecsvet.dbf')
im = "vecsvet.dbf"
DO pred with im
loop
endif

use vecsvet
         DO pros WITH kl
         CASE level3=3
kl=8                                 && Количество записей на 1 символ


if .not.file('vecfont.dbf')
im = "vecfont.dbf"
```

```
DO pred with im
loop
endif

use vecfont
         DO pros with kl
         CASE level3=4
kl = 15

if .not.file('vecproch.dbf')
im = "vecproch.dbf"
DO pred with im
loop
endif

use vecproch
         DO pros with kl
         CASE level3=5
SET COLOR TO W/N+,GR+/RB
close all
return
endcase
enddo
close all
return
Ж**ЖИсЛсЖ**#ЖЛс*Ж**Ж******Ж******<< End of prosk *****************************

Ф     xlz & ^k   Ф *4/ ^k & ^k   Ф xV   xb Uz xV   xb xV 4z ^k ^k  kp   Ф ^k ^k  v  ^k ^k Ц/ ^k xlz & ^k  ^k ^k   & ^k xV ^k ^k   Ф Ф Ф x^ 4k x^  4* n* 4k 4k 4k 4k 4*
Ж**********                    ГРОСМОТР                    ************
*  **** перед входом необходимо открыть базу и передать *********
*  ****        количество записей на один элемент              *********
*  ****                    kl                                  *********
i^x^^^x^xV^xlzxizxlz4z\lz^xiz^x^UzUkxlz\lzxlz\Vxlz^ж^xlz%^xVx^x^\llxxt\l/xU^xVx^^x^\^\^^^^^4k^x^^^^^4^^4k4k^4k4k^4k4k4k 4k 4k   T~ #*   T*



set color to w/n+
012,47,17,73 box                          && Закрасить место вывода сообщения
set color to gr+/b+,gr+/r

save screen to sere

DO WHILE .NOT. EOF()                      && до конца базы

set color to п/п
024,5 say "
set color to gr+/b+,gr+/r

024,27 say "ESC - ВЫХОД"
024, col 0+2 say "ENTER - Просмотр"
015,56 clear to 19,73
015,56 to 19,73 double
wshadow (15,56,19,73)
016,57 say 'В базе находится'
```

```
017,57 say (lastrec()/kl)
set color to gr+/b+
018,57 say ' элементов
07,36 clear to 19,50
07,36 say ""
1 = 0
DO WHILE Kio                        && вьводить по 10 записей
0row()+l,38 PROMPT char MESSAGE 'ESC-ВЫХОД , ENTER-просмотр'
i = i + 1
skip kl                             && количество записей на 1 элемент
enddo


asa = recno()                       && Запомнить положение указателя
0row()+l,37 PROMPT 'Ниже' MESSAGE 'ESC-ВЫХОД'
0row(),41 prompt 'BbllJe' MESSAGE 'ESC-ВЫХОД'
0row(),45 PROMPT 'ПОИСК' MESSAGE 'ESC-ВЫХОД'
set color to rb+/b+
07,36 to 20,50 double
wshadow (7,36,20,50)
set color to gr+/b+,gr+/r
MENU TO com
skip -(io*kl)                         && установка указателя в начало

if com = 0
restore screen from sere
release sere
use
close all
set color to w/n+,gr+/rb
return
ENDIF

.if com <11                           &&Локаз элемента
                skip (com*kl)-kl
                DO show WITH kl
                skip ~(com*kl)
                set cursor off
                restore screen from sere
                else

       if com=12
               skip -((com*kl)-(2*kl))          && Промотка вперед
               else

           if com =13                           && Поиск элемента
                    save screen to scr
               a = 0
               do while a=0

                    09,53 clear to 12,76
                    09,53 to 12,76 double
                    wshadow(9,53,12,76)
                    010,54 say 'Введите элемент поиска'
                    aa=spaee(ll)
```

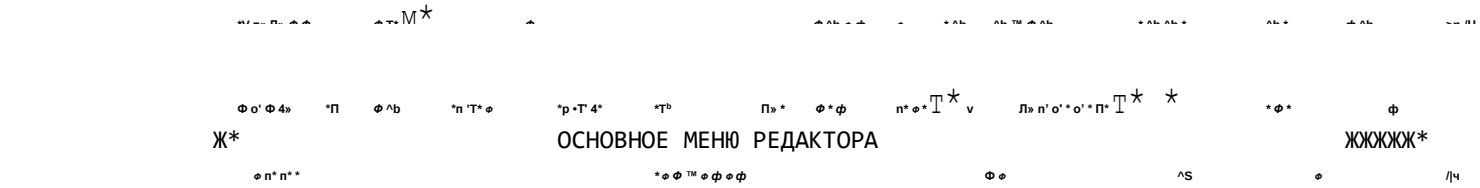
```
                    011,60 get aa
set cursor on
read
set cursor off
if aa=" "
«tsgl = ' Нет элемента . продолжать? '
do otvet
if otvet =1
set color to gr+/b+
loop
else
set color to gr+/b+
a = 1
endif
else
aa= rtrim(aa)
locate for char = aa
if .not. found()
msgl = "Элемент не найден.Продолжить?"
do otvet
if otvet = 1
set color to gr+/b+
loop
else
go asa ■
skip -(IOЖкI)
set color to gr+/b+
a = 1
endif
else
a = 1
endif
endif
enddo
«shrink (9,53,12,76,WnBorder,scr)
release scr
else
go asa
endif
endif
endif
ENDDO
restore screen from sere
release all
set color to g+/b
012,47 to 16,72
«shadow (12,47,16,72)
set color to gr+/b+
013,48 say " Записей больше "
014,48 say ”
015,48 say "              НЄТ
set                    color                     to                     w/n+,gr+/rb
use
RETURN
ЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖ« End of pros »ЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖ
```

зЖ iЖЖжк ж Ж Ж * Ж ЖжЖ * Ж ЖЖЖЖ Ж)jc Ж Ж Ж<Ж * * ЖЖж ЖЖж ЖжЖЖЖ Ж Ж ЖЖжЖЖЖ Ж
Ж)((Ж Ж Ж Ж Ж Ж Ж *Ж Ж * Ж Ж Ж Ж Ж Ж Ж ^Ж Ж ^Ж Ж Ж
жжжжж программа просмотра элементов библиотеки. жжжжжжж
жжжжж необходимо указать количество записей в базе на 1 элемент жжжжжжж

```
Tone (280,1)
setvideo(6)
Tone (130,1)
sethires(0)
clrscreen()
vel = 3                                   && Величина символов

boxfill (20,20,1290,960,BXNOFILL,GREEN)
boxfill (0,0,1330,1000,BXNOFILL,RED)
shade (5,5,0,1ight-red)
0 6,51  say "  F2 -  Увеличение "
0 8,51  say "  F3 -  Уменэшение "
010,51  say "  F4 -  Следующий элемент"
012,51  say "  F5 -  Предидущий элемент
014,51  say "  F10   - Выход 11
drawline (20,855,1320,855,0,0,5)
boxfill (820,20,15,835,0,light-red)
ЖЖr = picread (860,50,0,'com')


thi = 0
a = 0
do while a<>-9                              && Пока не нажата Flo
datareset()
02,5 say "Просматривается элемент: - "
set color to_g+/n
02,35 say char
set color to gr+/b+,gr+/r
f = 0
do while fOkl
                        datastore(xЖvel,yЖye1,move)
skip
f = f+1
enddo
select 2
   do vf with 100,120,0,1,2,3,thi,kl
select 1

a = inkey(0)
do case
case a = -1
vel = vel + .5
if vel = 4.0
- vel = vel - .5
endif
          case a    -2
```

```
          vel = vel - .5
if vel = 0
vel = vel + .5
endif
case a = -3
vel = 3
skip kl
case a = -4
vel = 3
skip -kl
ENDCASE
clrwin (25,25,810,850)
skip -kl
enddo
set cursor off
settext()
SET CURSOR OFF
skip kl
do zvuk
Tone (280,1)
RETURN
****************************«< End of show »>*******************************
Mr ^k     Φ     Φ Φ     ^k J# & ^k ^k & Φ ^k Φ & ^k Φ ^k ^k &     & ^k & & ^k     u?     ^k &     W ^k
»v *v *     * * * *     ™     w
**     VF     **
3* * 3* * * * * * * * * * * * * * * * 3* * * 2* * Ï* * 3* * * * * * * * * * * * * * * * * * * * * * * * * * 3* * * * 3* * * * 3|c * * * 3* * * * * * * * 3|c * 3* * *

*« VFDRAW.PRG
**
**
«* USAGE
**       do vfload with 1,'VECFONT'            &&must  first load stroke font-data
**
**       x = 675                               &&x and  y origin of string
**       y = 500
**       ang = 45                              &&angle  of string countei—clockwise
**       length = 200                          &&длина  строки на экране
**       Imode = 0                             &&(OR,   XOr etc.)
**       colour = 4                            &&цвет
**       string = '3210'                       &&строка для рисования
«*
** do vf with x,y,ang,length,Imode,colour,string
**
»*

parameter x,y,ang,length,Imode,1clr,this,kl
private rad,normwid,width,chnum,baseindx
normwid = 100+35                               && normal char width = 100 + 35 (gap)
r = fixpos(x,y)                                && CLIPPER, FOX etc.
nchars =1
width =1
chnum = 1
baseindx = this*kl                             && 8 strokes per char
```

```
   rad = (chnum-1)*width
  r = datarange(baseindx,baseindx+(kl-l))             && CLIPPER, FOX etc.
  r = po1yvec(rad,ang,1,ang,Imode,0,1 c1r) && CLIPPER, FOX etc.
return
```




```
SAVE SCREEN TO SCR
Level3=6
DO WHILE.NOT.level3=0
SET COLOR TO GR+/B+
614,15 TO 20,41 DOUBLE
wshadow (14,15,20,41)
SET COLOR TO G+/B+
615,16 PROMPT    ’    СТРЕЛКИ
616,16 PROMPT    '    СВЕТОФОРЫ
617,16 PROMPT    '    ШРИФТ
618,16 PROMPT    ‘    ПРОЧЕЕ
619,16 PROMPT    '    ВЫХОД
MENU TO leve!3

DO CASE
       . CASE level3=1

if .not.file('vecstrel.dbf')
im = "vecstrel.dbf"
DO pred with im
loop
endif

use vecstrel
kl = 15
naz = "С ТР Е Л К И"
         DO vfdemo with kl,naz
restore screen from scr
         CASE level3=2

if .not.file('vecsvet.dbf ')
im = "vecsvet.dbf"
DO pred with im
loop
endif

use vecsvet
kl = 20
naz = "С В Е Т О Ф О Р"
         DO vfdemo with kl,naz-
restore screen from scr
         CASE level3=3
```

```
if .not.file('vecfont.dbf')                                    '
im = "vecfont.dbf"
DO pred with im
loop
endif
use vecfont
kl = 8
naz = ”Ш Р И Ф Т"
DO vfdemo with kl,naz
restore screen from scr
CASE leve!3=4
if .not.file('vecproch.dbf')
im = "vecproch.dbf"
DO pred with im
loop
endif
use vecproch
kl = 15
naz = "ПРОЧ Е Е"
DO vfdemo with kl,naz
restore screen from scr
CASE level3=5
SET COLOR TO W/N+,GR+/RB
RESTORE SCREEN FROM scr
close all
return
ENDCASE
ENDDO
```

^k ^k ^k ^k ЕГ м >4 -y ^k ^k ^k ^k ^k ^k & ^k ^k & & ^k & ^k ^k ^k
^P ^P ^p ^p ^p ^p ^P ^p II Ш I I' |J* ƒ ^p ^P ^P ^P ^P ^p ^P ^P ^P ^p ^P ^P ^P ^P ^P ^P ^p ^P ^P ^P ^p ^P ^P ^P ^P ^p ^P

1 3^C)^C J^C j|c *)|t)|( )|c ж)|C ^|C ^jc)$(*)|c *)jc)|( )|c)fC)|(*)|C *)|( )^C)|(Xc)|(^|c)|C *)|(*)j(1|C)|C)|C)|C *)|C)|c *)|C }|c)|t)|(# JfC ?|C)|( )ft)|t)|c)|c ?|c * * * * *)^C j|t

```
**   VFDEMO.PRG                                *****
2   *                                  ОБОЛОЧКА ДЛЯ РЕДАКТОРА                                  *****
3   *                      УКАЗАТЬ БАЗУ И КОЛИЧЕСТВО ЗАПИСЕЙ НА ЭЛЕГЕНТ                        *****
```

%v къ %ъ <ъ <ъ o/ & ф ^к ^к Мк ^к ^к ^к ^к
ф ф ф Ж Ж Ж ф ф * ф п» * * п* * л» * л* * п» *

```
private vfdir
setvideo(6)
sethires(O)
loadcset(0,"int1437s")
clrscreen()
DO ZVUK
clrwin (680,510,1000,700)

retcode = pcxSD(pcxtype)
retcode = pcxSM(pcxGRAPHICS)
DO FRAME with naz
DO FRAME1
set color to GR+/N
Є 24,32 say 'F1-'
в 24,40 say 'F2-'
```

```
            e 24,62 say   'F4-'
            e 24,71 say   'F7-'
            set color to  R/N
            & 24,35 say   "Help"
            0 24,43 say   "Просмотр
            0 24,55 say   "начало"
            0 24,65 say   "конец"
            Q 24,74 say   "ПОИСК"

            saystring(850,120,0,0,12,                              "Fl                                  0-ВЫХОД")
saystring(40,60,0,0,10,'PgUp/PgDn        '+chr(27)+'        '+chr(24)+'        '+chr(25)+'        '+chr(26)+
.РЕМЕЩЕН/Е [Enter3-РЕДАКТИРОВАНИЕ')
            **r = picread (1040,20,0,'comp')
            box-fill (10,10,1330,980,BXNOFILL,GREEN)
            boxfill (0,0,1350,1000,BXNOFILL,RED)
            shade (5,5,0,1ight-red)
            boxfill (10,156,515,563,BXNOFILL,GREEN)

            DO EDIT with kl

            settext()
            pcxSM(pcxTEXT)
            set                                        cursor                                        off
            do                                                                                      zvuk
            SET                        COLOR                        TO                        R/BG+,GR+/RB-
            RESTORE                          SCREEN                          FROM                          sc
            return
            4   ** * ж* *** * ** * * ж *** ****** ЖЖ **<< End of fvdemo >>*****************************



            5   *                  EDIT                      m
            jj' * * * *   * * i* * ** * * * * * * * Ї* * )* * * ijc )(c * э|( * * * * )(c * i*
            * s(c

            select 1
            DECLARE Fields[43
            FieldsCl] = "Char"
            FieldsE2] = "x" .
            Fields[33 = "y"
            Fields[43                                      =                                      "move"-
            SET                          COLOR                              TO                          W/N,GR+/N
            DBEDIT(7,1,18,30,Fields,"UserFun")
            use
            close                                                                                       all
            return


            ***************************************************************************
            6   ***************<<    пользовательская    функция    для    DB-Edit    >>******************
            ***************************************************************************


            FUNCTION USERFUN
```

```
PARAMETERS mode,fld_ptr
PRIVATE cur_field
cur_field = fieldsCfld_ptr]
DO CASE
CASE mode<4
RETURN 1
CASE LASTKEYO = 28                          && Fl- HELP
pcxtype = pcxVGA_12
retcode = pcxSD(pcxtype)
retcode = pcxSM(pcxGRAPHics)                && Установка режима графики
bufmax = 20000
buffer = space(bufmax)
bufsize = pcxDB(buffer,bufmax,325,49,610,325,0)
pred = 0
PUBLIC var
set color to GR+/N
if .not.file(.'help.txt')
boxfill (750,500,495,200,0,red)
boxfill (750,500,495,200,bxnofi 11,cyan)
@10,50 say "НЕТ ФАЙЛА help.txt"
@13,49 say "Нажмите любую кнопку"
inkey(0)
pred = 1
endif
if pred = 0
var = MEM0READ('help.txt')
boxfill (685,325,630,595,0,cyan)
var = MEM0EDIT(var,3,42,19,76,.f.)
endif


clrwin(685,323,1320,920)
retcode = pcxBD(buffer,bufsize,325,49,0)
set color to W/N _
DO FRAME1
RETURN 1
         CASE LASTKEYO =     -9             && F10 - ВЫХОД
         RETURN 0 .
         CASE LASTKEYO = -7
                                             &&F8 -
         0 24,4 SAY "СИМФЕРОПОЛЬ"
         RETURN 1
         CASE LASTKEYO       -2             &&F3 ~ НАЧАЛО
         GO TOP
         RETURN 1
         CASE LASTKEYO       -3             &&F4 -  КОНЕЦ
         GO BOTTOM
         RETURN 1
         CASE LASTKEYO       -6             &&F7 -  ПОИСК
         A = space(15)
b = char
boxfill (75,750,340,136,0,RED)
boxfill (75,750,340,136,BXNOFJLL,CYAN)
         set color to gr+/n,w/n
         @4,6 say "Введите название"
         05,6 say " элемента "
```

```
         set color to r+/n,w/n
         e 6,6 say"*"GET A
         READ
         л = rtrim(a)                    && Убрать пробелы справа
         set color to gr+/n,gr+/n
         LOCATE FOR CHAR=A               && Установить указатель на элемент
if found()

a 4,6 say "Найдена      запись"
a 5,6 say "
else
a 4,6 say " не найден "
a 5,6 say " элемент
set color to g+/n,gr+/n
Є 6,8 say A
locate for.Char = b                      && Восстановить прежнее состояние
inkey(O)
endif
clrwin (75,750,415,895)
set color to w/n,gr+/n
RETURN 1
CASE LASTKEYO = -1                       && F2 - Просмотр
Clrwin (687,327,1300,900)
a 1,55 SAY CHAR
thi = 0
do FRAHEl
r - datareset()

f = 0
do while fOkl
r = datastore(x*3,y*3,move)
skip
f = f+1
enddo
select 2
  do   vf   with   690,328,0,1,2,3,thi,kl
select                                1
skip                                 -kl
RETURN 1
         CASE LASTKEYO = 13              && ENTER
         SET COLOR TO W/N,R+/N
         aR0W(),C0L() GET &cur_field
         READ
         SET COLOR TO W/N,GR+/N
         KEYBOARD CHR(4)
         RETURN 1
ENDCASE
         RETURN 1

t****O******************<< End of       userfunction >>*Ж*Ж**Ж*******ЖЖ*5К*ЖЖ*Яс»ЖЯ<*
```

```
          Mr W 4                                & Mf          & ^И          & ^f *^ ^f Ф ^к & Ф &          ^к          & Ф J>
     ф *Т*   *т»                        ^S ^^                      Ф                                                    ^ч
**                        ОСНОВНОЕ МЕНЮ УДАЛЕНИЯ                                          ******
```

```
SAVE SCREEN TO SCR

leve!3=6
DO WHILE.NOT.Level3=0
SET COLOR TO GR+/B+
в14,15 TO 20,41 DOUBLE
wshadow (14,15,20,41)
SET COLOR TO G+/B+
«15,16 PROMPT '      • ’.СТРЕЛКИ
016,16 PROMPT ' СВЕТОФОРЫ
017,16 PROMPT    '    ШРИФТ
018,16 PROMPT    '    ПРОЧЕЕ
019,16 PROMPT    '    ВЫХОД
MENU TO 1eve13
DO CASE
         CASE leve!3=l

if .not.file('vecstrel.dbf ')
im = “vecstrel.dbf"
DO pred with im
loop
endif

use vecstrel
kl = 15
         DO DELETE WITH kl
         CASE level3=2

if .not.file( vecsvet.dbf')
im = "vecsvet.dbf“
DO pred with im
loop
endif

use vecsvet
kl = 20
         DO DELETE WITH kl
         CASE level3=3

if .not.file('vecfont.dbf')
im = "vecfont.dbf"
DO pred with im
loop
endif

use vecfont
kl = 8
         DO DELETE WITH kl
         CASE leve!3=4
```

```
 if .not.file('vecproch.dbf')
im = "vecproch.dbf"
DO pred with im
loop
endif

 use vecproch
kl = 15
          DO DELETE WITH kl
CASE level3=5
SET COLOR TO W/N+,GR+/RB
RESTORE SCREEN FROM scr
CLOSE ALL
return
ENDCASE
ENDDO
```

```
                                PROCEDURE delete
do while .t.
                          = space(ll)

                          Є9,45 clear to 15,71
                          09,45 to 15,71
                          wshadow (9,45,15,71)
                          set cursor on
                          011,46 say "Введите название элемента"
                          @12,54 get a
                          read

                          set cursor ff

                          aa = 1
                          if a = " "
                          aa = 0
                          endif

                        if aa = 0
                        msg2 = " нет названия "
                        msgl = msg2+".ГрОДОЛЖать? "
                        do otvet
                        if otvet = 1
                        set color to g+/b+
                        loop
                        else
                        wshrink (9,45,15,71,WnBorder,scr)
                        close all
                        return

                           endif
```

```
                    endif
locate                    for                    char                    =                    a
if                                                                                          found()
.         msgi         =         "         запись         существует,         удалить?         "
do                                                                                          otvet
if                             otvet                              =                             2
set                            color                          to                            g+/b+
loop
else
z=0
do                                       while                                              zOkl
delete
skip
z                        =                        z                        +                        1
enddo
pack
msgl=                    'Элемента                    больше                    нет.Продолжать?'
do                                                                                          otvet
if                                          otvet                                             =1
set                            color                          to                            g+/b+
loop
else
«shrink                                                                  (9,45,15,71,WnBorder,scr)
close                                                                                       all
return
endif
endif
else
msg2               =               "               не               найдена               запись"
msgl                   =                  msg2+"                  .Продолжать?                  "
do                                                                                          otvet
if                             otvet                              =                             1
set                            color                          to                            g+/b+
loop
else
«shrink                                                                  (9,45,15,71,WnBorder,scr)
close                                                                                       all
return
endif
endif
enddo
Ж#ЖЖ*ЖЖ*******ЖЖ**Ж**Ж**<<< End of delete »>*****************************
```

```
ф Це&&&&&ïï/Ф&&&&&&&&&&iП/&&                    &&&&&          &to&Ф&&&&Ф&&&&&&&&&&&&Ф
                                                  ф Ф      ф ф                    Ф         П» *V Ф Т*
**                        ОСНОВИН МЕНО ДОПОЛНЕНИЯ                                          ******
ф       tL   ф  Це Це d/&       ф       <Ъ «Ъ          ф        «D «Ъ & ib ₒ       & & U/ &ₒ Ф J/
                                Ф       ™
```

```
level3=6
DO WHILE.NOT.level3=0
SET COLOR TO GR+/B+
```

```
614,15 TO 20,41 DOUBLE
wshadow (14,15,20,41)
SET COLOR TO G+/B+
615,16 PROMPT '        СТРЕЛКИ
616,16 PROMPT '        СВЕТОФОРЫ
617,16 PROMPT     ‘    ШРИФТ
618,16 PROMPT     *    ПРОЧЕЕ
619,16 PROMPT     '    ВЫХОД
MENU TO level3
DO CASE
          CASE level3=l

if .not.file('vecstrel.dbf')
im = "vecstrel.dbf"
DO pred with im
loop
endif

kl = 15
use vecstrel
          DO dopl with kl
          CASE level3=2

if .not.file('vecsvet.dbf')
im = "vecsvet.dbf"
DO pred with im
loop
endif

kl = 20
use vecsvet
          DO dopl with kl
          CASE level3=3

if ,not.file('vecfont.dbf')
im = "vecfont.dbf"
DO pred with im
loop
endif

kl = 8
use vecfont
          DO dopl with kl
          CASE level3=4

if .not.file('vecproch.dbf')
im = "vecproch.dbf"
DO pred with im
loop
endif

kl = 15
use vecproch
          DO dopl with kl
          CASE level3=5
```

```
SET COLOR TO W/N+,GR+/RB
release all except level2
close all
return
ENDCASE
ENDDO
«#******)K****>K>K******#*<< End of dop. >>***Ж****iK***iк**$***Ж****iK***Ж)K*****)C
```

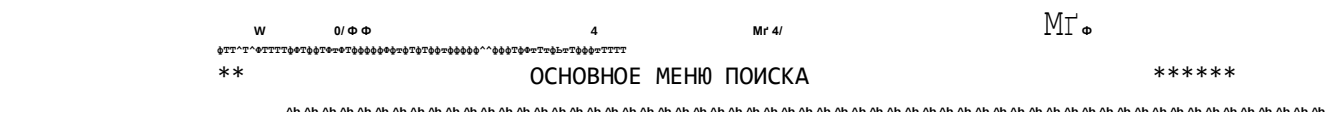
^P ^P *R ^P ^P * ^P ^P ^P ^P Mr & iD & & Mr & Mr Mr Ф Mr & Ф 'A* Mr Ф Mr & Ф Mr

```
Ж»Ж****ЖЖ                ДОБАВЛЕНИЕ ЭЛЕМЕНТОВ В БАЗУ                **********
*********                      открыть базу                      **********
Ж*#****»* указать количество записей на один элемент - kl **********
```

«Ь kV tb U» «Ь *Ъ »Ь Л⁄л ф «Ъ ф kG k^ -£> к^ «Ъ Uf к^ «G *Ъ Ф Ф

```
save screen to sere
go top
msg2 = 'Продолжать?'
DO WHILE .T.
char_t = SPACE(ll)
set color to gr+/g
09,39 clear to 14,71
09,39 to 14,71 double
wshadow (9,39,14,71)
set cursor on
set color to r/g
0 10,44 say "Введите название элемента"
0 12,51 get char_t
read
set cursor off
if char_t = " "
                     msgl = ' нет названия .'+msg2
                do otvet
                if otvet = 1
                     loop
        else
        restore screen from sere
        return
        endif
        endif
char_t = RTRIM(char_t)

msgl = ' хотите внести изменения ?
DO otvet
          IF otvet = 1
          LOOP
          ENDIF

LOCATE FOR char = char_t
IF .NOT. FOUND()
move_t =1
APPEND BLANK
REPLACE char WITH char_t,move WITH move_t
a = 0
```

```
  do while a<kl-l
append blank
REPLACE move WITH move_t
a = a+1
enddo

  msgl = * Записано ! '+msg2
DO otvet
IF otvet = 2
set color to g+/b+
restore screen from sere
return
ENDIF
ELSE

  msgl = "Название существует."+msg2
DO otvet
IF otvet = 2
restore screen from sere
return
else
loop
endif

  ENDIF
ENDDO

Ж*****Ж**ЖЖ**ЖЖЖ5(СЖ**Ж*ЖЖ*<<< End of dopl »>ЖЖЖЖЖЛсЖ***ЖЖ**Ж**ЖЖЖ5КЖЖЖ*Ж**Ж
```




```
  SAVE SCREEN TO SCRE
level3=6
DO WHILE.NOT.leve!3=0
SET COLOR TO GR+/B+
014,15 TO 20,41 DOUBLE
wshadow (14,15,20,41)
SET COLOR TO G+/B+
015,16 PROMPT '        СТРЕЛКИ
016,16 PROMPT '        СВЕТОФОРЫ
017,16 PROMPT    '     ИРИФТ
018,16 PROMPT    '     ПРОЧЕЕ
019,16 PROMPT    '     ВЫХОД
  MENU TO level3
DO CASE
CASE leve!3=l

  if .not.file('vecstrel.dbf')
im = "vecstrel.dbf"
DO pred with im
loop
```

```
endif

          use vecstrel
KL = 15
          DO POIS WITH KL
          CASE level3=2

if .not.file('vecsvet.dbf')
im = "vecsvet.dbf"
DO pred with im
loop
endif

          use vecsvet
kl = 20
DO pois with kl
CASE level3=3
if .not.ffle('vecfont.dbf')
im = "vecfont.dbf”
DO pred with im
loop
endif
use vecfont
KL = 8
DO POIS WITH KL
CASE level3=4
if .not.filet'vecproch.dbf')
im = "vecproch.dbf"
DO pred with im
loop
endif
use vecproch
kl = 15
DO pois with kl
CASE level3=5
SET COLOR TO W/N+,GR+/RB
RESTORE SCREEN FROM sere
close all
return
ENDCASE
ENDDO
RETURN
Ф   Ж Ж     Ж ^Р ₒ ₒ Ж ® ®Ж ®                ® ^p        v ^p        ®       W ^p   v      ^p
o*
procedure pois

016,54 clear to 20,71
016,54 to 20,71 double
wshadow (16,54,20,71)
017,55 say 'В базе НЗХОДИТСЯ'
```

```
set color to r/b+
€18,55 say (lastrec()/kl)
set color to gr+/b+
€19,55 say ' ЗЛЄМЄНТСВ
do while .t.
a = space(ll)
€10,45 clear to 13,71
€10,45 to 13,71
wshadow (10,45,13,71)
set cursor on
011,46 say "Введите название элемента"
012,54 get a
read
set cursor ff
aa = 1
if a = " "
aa = 0
endif
a = rtrim(a)
locate for char = a
if found()
msg2 = " Найдена запись "
else
msg2 = " Не найдена запись"
endif
if aa = 0
msg2 = " нет названия
endif
msgl = msg2+".Продолжать? "
do otvet
.if otvet = 1
set color to g+/b+
loop
else
close all
restore screen from sere
set cursor off
return            •
endif
enddo
*************************<< End of poisk ***********************************


ffr W &              & ^k ^k ^k ^k ^k              ^k Ф       Ф^k ^k ^k ^k & ^k ^k       ^k ^k ^k ^k ^k & ^k ^k ^k ^k              ^k ^k ^k ^k ^k ^k ^k ^k ^k      ^k & & Uk Mk ^k ^k ^k
  ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft ft    ft    ft ft    ft ft ft ft ft ft ft ft ft ft ft ft ft Ф ft ft    ft ft ft ft ft ft ft ft ft ft ft   ft ft ft   ft ft ft ft
**                                PROCEDURE spravka                                **
^k ^k W & &          ^k ^k ^k & ^k W ^k ilr ^k ^k ^k ^k        Ф   ^k     Ф Ф Ф & ^k   ^k  Ф ^k & ^r ^k ^k ^k &          ^k ^k   4       ^k ^k Mk
ft ft ft ft ft ft ft ftft ft ft ft ft ft ft ft ft ft ft ft ft              ft    ft ft ft


SET COLOR TO R+/B
012,26 TO 20,71 DOUBLE
wshadow (12,26,20,71)
```

```
SET COLOR TO W+/B
€13,27 SAY '          ПРОГРАММА РАЗРАБОТАНА НА СУБД
<214,27 SAY "              CLIPPER SUMMER'87,
€15,27 SAY '              С ИСПОЛЬЗОВАНИЕМ
€16,27 SAY '            ПАКЕТОВ DGE И PCX,
Tone(280,l)
INKEY(O)
SET COLOR TO R/BG
RETURN
Ж*Ж*****Ж*************ЖЖ<< End of spravka >>**##*WM***O*M*W**)K*********

Лф     ******ф              J.    Ja         kb.ЬФФ      ФФФU>Ф Ф        ФФ          Ф
***                        ОСНОВНОЕ МЕНЮ СЕРВИС                               ****


SAVE SCREEN TO SCR


level3=6
DO WHILE.NOT.leve!3=0
SET COLOR TO GR+/B+
€9,38 TO 14,54 DOUBLE
wshadow (9,38,14,54)
SET COLOR TO G+/B+
010,39 PROMPT 'КАЛЬКУЛЯТОР
€11,39 PROMPT 'КАЛЕНДАРЬ
€12,39 PROMPT 'ЗАПИСНАЯ КНИЖКА'
€13,39 PROMPT 'ВЫХОД
MENU TO leve!3
DO CASE
         CASE leve!3=l
Tone (280,1)
set color to rb+/b+
.€9,6 to 14,31 double
wshadow(9,6,14,31)
set color to g+/b+
€10,7 say "калькулятор поставляется"
€11,7 say " за отдельную плату,
set color   to gr+/b+
€12,7 say   " ВСЄГО       10$
set color   to g+/b+
€13,7 say   " И ОН     ВЭШ.            "
inkey(O)
t**run cal.exe
restore screen from scr
set cursor off
         CASE level3=2
memory = memory(O)
*t€l,l say memory
if memory >115

if .not.file('kal.exe')
i m = " ka 1. e x e "
DO pred with im
loop
```

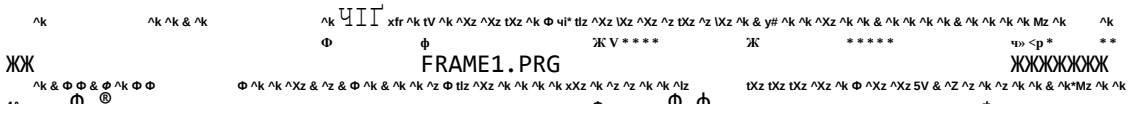
```
endif
run kal.exe
else
set color to gr+/r
69,6 to 14,32 double
wshadow (9,6,14,32)
eio,7 say " У вашей машины в памяти "
611,7 say " столько хлама, "
612,7 say "что нет места для запуска"
613,7 say " календарика. "
inkey(0)
set color to g+/b+
endif
restore screen from scr
set cursor off
CASE level3=3

if. .not.file('help.dbf')
im = "help.dbf"
DO pred with im
loop
endif

use help

z = "blocnot"

locate for topic = z

msgl = ' Убрать предидущие записи ? '                                    '
do otvet
if otvet = 1
x = " "
replace topic_text with x
endif

69,9 clear to 20,70
set color to r+/b+
69,9 to 20,70 double
wshadow (9,9,20,70)
set color to gr+/b+
69,32 say 'Записная книжка'

620,20 SAY 'Последняя запись была сделана: - '

620,53 say LUPDATEO
set color to п/п
624,5 say "
set color to gr+/b+
624,0 say "Esc-ВЫХОД"
624,col()+2 say "Ctrl-W-СОХранить И ВЫЙТИ"
624,col()+2 say "Ctrl-N-ВСТав. строку"
624,col()+3 say "Ctrl-Y- удал. строку"
set cursor on                      .       .
set color to g+/b+

replace topic_text with memoedit (topic_text,10,10,19,69,.t.)
```

```
close all
set cursor off
restore screen from scr
CASE level3=4
SET COLOR TO W/N+.GR+/RB
CLOSE ALL
RESTORE SCREEN FROM SCR

wshrink (9,38,14,54,WnBorder,scr)
restore screen from scr
return
endcase
enddo
restore screen from scr
release scr
return
*Ж*Ж*ЖЖ***Яс**********9|oK*Ж<< End of servis >>*******************:*************
```

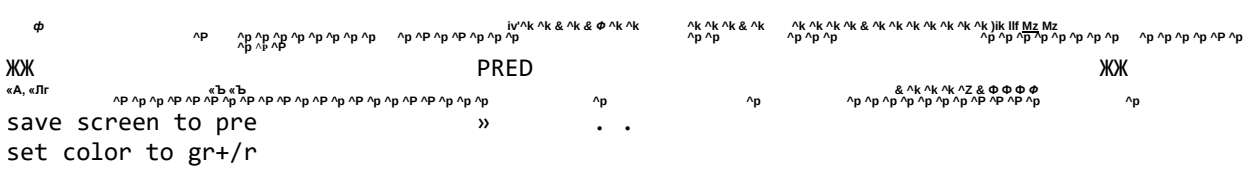
^t 3(t jfc з|(ЖЖ Ж 3(t 30c * jjt Ж jft ж зк 3(t jft 3fc 3jt з^cift3ft 3ft э|c 3jt)jt)jt jft 3ft3ft jft 3fc JftJftЖ Jft Jft jft 3fc jft 3fc Jft з|c Ж Jft Jft Jft Ж Jft Ж 3ft Jft Jft Ж Ж Jft Jft

ЖЖ FRAME.PRO **ЖЖЖ**

```
r = boxfill(495,235,835,695,BXNOFILL,CYAN)              && boxes
r = boxfill(485,225,853,713,BXNOFILL,GREEN)
shade (490,230,0,GREEN) •
r « boxfill(55,915,385,65,0,blue) ,
r = B0XFILL(55,915,385,65,BXNOFILL,RED)
SET COLOR TO R+/N
01,9 say naz
SET COLOR TO W/N
return
```




```
r = xyaxeэ(685,325,615,600,20,20,0+4Ж2+16Ж2+'64,дг&y)
st = '    10 20 30 40 50 60 70 80 90 1001101201301401501601701801902000'
r = labelx(705,300,31,3,1,0+1+8+2,white,st)
st = '0 10 20 30 40 50 60 70 80 90 100110120130140150160170180190'
r = labely(660,305,31,3,1,1+1+8,white,st)
RETURN
ЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖ« End of framel »ЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖ.ЖЖЖЖЖЖЖЖЖЖЖЖЖЖ
```




```
save screen to pre                  »        . .
set color to gr+/r
Tone (280,1)
014,35 clear to 16,70               .
```

```
€14,35 to 16,70 double
wshadow(14,35,16,70)
€15,40 say "HET Ф/WIA
set color to bg/r
€15,col()+l say im

inkey(O)
set color to gr+/b+
restore screen from pre
return

ЖЖ                          PROCEDURE otvet                                              ЖЖ

PUBLIC otvet
save screen to s
otvet=2
SET COLOR TO GR+/R,RB/G,,,W/N
€13,22 CLEAR TO 17,56
€13,22 TO 17,56 DOUBLE
wshadow (13,22,17,56)
€13,24 SAY ' '+msgl+' '
€17,27 PROMPT ' ДА '
€17,50 PROMPT 'HET '
MENU TO otvet
SET COLOR TO W/N+,GR+/RB
€13,22 CLEAR TO 17,56
restore screen from s
release s
RETURN
Ж*****************Ж*Ж*<< End of otvet

**                              PROCEDURE zvuk                                           **
T0NE(220,l)
T0NE(277.20,2)
RETURN
tOM.WKM*****************^ End of zvuk >>******************************

* Function: WEXPAND(<yl>,<xl>,<y2>,<x2>,<b>)                                             *
* Развертьвание окна                                                                     *
Ж        Параметры: 1-4 - координаты окна, 5- символы рамки (8-9 шт) ж

FUNCTION WEXPAND
PARAMETERS Y1,X1,Y2,X2,Brdr -
IF WnExpand
PRIVATE I
   Ж эффект развєртьвания
```

```
    FOR I=INT((X2-X1)/2-l) TO O STEP -2
        Є Y1,X1+I,Y2,X2-I BOX Brdr
    NEXT
    * сигнал
    If WnSound
        Tone(280,l)
    EndIF
ENDIF
Є Y1,X1,Y2,X2 BOX Brdr
IF WnShadow
    * тень
    WShadowtYl,X1,Y2,X2)
ENDIF
RETURN .T.
************************ End of WEXPANDO *********************************

*********************************************************************
* *
*  Function: WSHAD0W(<yl>,<xl>,<y2>,<x2>)                          ‘          *
*  Рисует прозрачную тема справа от заданного окна                           *
*  переработанный для '87 вариант функции SHADOW из Clipper 5 *
*********************************************************************

FUNCTION WShadow
PARAMETERS Y1,X1,Y2,X2
PRIVATE Nyl,Nxl,Ny2,Nx2
Nyl=MIN(Y2+1,24)
Ny2=Nyl+l
Nxl=Xl+l
Nx2=MIN(X2+1,79)
RESTSCREEN! Nyl, Nxl, Ny2, Nx2,;
    TRANSFORM! SAVESCREEN(Nyl, Nxl, Ny2, Nx2),;
    REPLICATE!"X", Nx2 - Nxl + 1 )+REPLICATE("XX",Nx2-Nxl+l) ) )
,Nyl=Yl+l
Ny2=Y2+l
Nx 1=MIN(X2+1,79)
Nx2=Nxl+l
RESTSCREEN( Nyl, .Nxl, Ny2, Nx2,;
    TRANSFORM! SAVESCREEN(Nyl, Nxl , Ny2, Nx2),;
    REPLICATE!"XXX'', Ny2 - Nyl + 1 ) ) )
RETURN 0
*************************** End of WSHADOW!) *************************

*********************************************************************
* Function: WShrink(<Yl>,<Xl>,<Y2>,<X2>,<Border>,<Screen>)                *
*         Сворачивание окна                                               *
t         Параметры: 1-4 - координаты окна, 5 - символы рамки,            *
*         6 - образ экрана                                                *
*********************************************************************

FUNCTION WShrink
PARAMETERS Y1,XI,Y2,X2,Brdr,Savscr
IF WnShrink
PRIVATE I
    * эффект сворачивания
```

```
                                                    REST SCREEN FROM Savscr '
                                                FOR 1=0 TO INT((X2-X1)/2-l) STEP 2
                                                        Є Y1,X1+I,Y2,X2-I BOX Brdr
                                                           REST SCREEN FROM Savscr
                                                                              NEXT
                                                                         * сигнал
                                                                      If WriSound
                                                                      Tone(200,l)
                                                                            EndIF
                                                                             ELSE
                                                           REST SCREEN FROM Savscr
                                                                            ENDIF
                                                                       RETURN .T.
********************** End of WShrinkO ****************************
```

ТЕКСТ ПРОГРАММЫ ФОРМИРОВАНИЯ СТАТИЧЕСКОЙ МОДЕЛИ ЖЕЛЕЗНОДОРОЖНОЙ СТАНЦИИ

```
ЖЖ ’ ОСНОВНАЯ ПРОГРАММА СОЗДАНИЯ СТАТИЧЕСКОЙ МОДЕЛИ Ж.Д. ЖЖЖЖ
ЖЖ                                СТАНЦИИ                                    ЖЖЖЖ
1Ж Ж Ж Ж * Ж Ж Ж Ж Ж Це Ж Ж Ж Ж Ж )(с Ж Ж Ж з(с Ж jjc Ж Ж Ж Ж sfe Ж эфе Ж)((Ж
```

```
00,0 clear
set talk off
set echo off
set status off
set bell off
set scoreboard off
set console off
set cursor off
set wrap on
set procedure to pexlib
SET PROC TO _wshadow

PUBLIC WnExpand,WnShrink,WnShadow,WnBorder,WnBackgr,WnSound
public vptr,scr,new,NAME,kart

 WnExpand=.t.                      && Раскрытие окон
 WnShrink=.t.                      && закрытие окон
 WnShadow=.t.                      && тени у окон
 HnBorder= "——llUsjtl"             && Рамки окон
 WnBackgr="^''                     && символ-заполнитель
 WnSound=.T/                       && сигналы при раскрытии/свертке

 NAME = SPACE(8)+".PCX"
 kart = O
 new = O


 if .not.file('DGEDEFS.PRG')          && public defines
   do ..\DGEDEFS
 else
 do DGEDEFS
 endif

 do pcxlnit

 pcxtype = pcxEGA_10
 retcode = pcxSD(pcxtype)
 retcode = pcxSM(pcxGRAPHICS)
 pcxSP(l)

 pict = "zas.pcx"

 pal = SPACE(17)
 retcode                          =                          pcxGFP(pcxtype,pict,pal)
 retcode = pcxSDP(pal)
 retcode = pcxFD(pict,0,0,0)

 set color to r+/n
 02,6 say date()
 02,64 say time()
 pcxSP(O)
```

```
inkey (0)

pcxSM(pcxTEXT)
set cursor off
DO zvuk
60,0 CLEAR
SET COLOR TO gr+/b+
60,0 CLEAR TO 23,79
60,0,23,79 box '.—.0=1111'
SET COLOR TO R+/IM+
60,15 TO 2,62 DOUBLE
SET COLOR TO G+/N+
61,16 SAY '          РЕДАКТОР ЖЕЛЕЗНОДОРОЖНЫХ СТАНЦИИ
DO WHILE.T.
         SET COLOR TO gr+/BG,GR+/RB
         64,3 TO 6,29 DOUBLE
         wshadow (4,3,6,29)

         64,33 TO 6,48 DOUBLE
         wshadow (4,33,6,48)

         64,52 to 6,61 double
         wshadow (4,52,6,61)

         64,65 TO 6,75 DOUBLE
         wshadow (4,65,6,75)

         SET COLOR TO R/BG,GR+/RB
         65,4 PROMPT " РЕДАКТИРОВАНИЕ СТАНЦИИ "
         65,34 PROMPT "СОЗДАНИЕ НОВОЙ"
         65,53 PROMPT " HELP "
         65,66 PROMPT " ВЫХОД "
         MENU to level1

DO CASE
CASE levell=l
save screen to scr
new = 1
DO Id
restore screen from scr
CASE levell=2
save screen to scr
new = 0
DO resmous
DO start
restore screen from scr
CASE levell=3
do help
**do iconshde
CASE levell=4
msg 1='ВЫ УБЕРЕШ, ЧТО ХОТИТЕ ВЫИТИ ?'
DO otvet
```

```
IF otvet=2
loop
ENDIF

DO zvuk
CLEAR
set color to w+/b+
69,20 clear to 20,60
69,20 to 20,60 double
610,22 say '
611,22 say '
612,22 say '           конец работы
613,22 say '           .
614,22 SAY '
clear all
close all
SET TALK ON
SET BELL ON
SET STATUS ON
CANCEL
ENDCASE
ENDDO

  clear
quit


** ПРОЦЕДУРА ВВОДА ИМЕНИ РЕДАКТИРУЕМОЙ СТАНЦИИ **
* * * * * * * * * * * * * * * * *                    L. D   * )jc


dd = O
aq = 1
.do while aq =1
614,22 clear to 17,56
614,22 to 17,56 double
wshadow(14,22,17,56)
615,23 say "ВВЕДИТЕ ИМЯ РЕДАКТИРУЕМОЙ СТАНЦИИ"
616,33 GET name
set cursor on
read
name = rtrim(name)
name = upper(name)
if name = " "
dd = 1
msgl=" НЕТ ИМЕНИ ФАЙЛА.ПОВТОРИТЬ ? "
DO OTVET
IF OTVET =2
return
endif
else
aq=0
endif
set color to r/bg,gr+/rb
if dd = 0
```

```
if f ile(name) = .F.
msgl=' НЕТ ФАЙЛА '+NAME+'.'
€19,33 to 21,44
€20,34 say 'ПОВТОРИТЬ?'
DO OTVET
if otvet = 2
set color to b/b,b/b
€19,33 clear to 21,44
€19,33 to 21,44
return
else
set color to b/b,b/b
€19,33 clear to 21,44
€19,33 to 21,44
aq=l
endif
endif
dd=O
endif
set color to r/bg,gr+/rb
dd=O
enddo

DO LOAD


** ПРОЦЕДУРА ЗАГРУЗКИ РЕДАКТИРУЕМОЙ СТАНЦИИ ***


do resmous
set cursor off
setvideo(6)
setver(2)
clrscreen()
pal=SPACE(17)
pcxtype = pcxEGA_10
retcode = pcxSDP(pal)
retcode = pcxSM(pcxGRAPHICS)
sethires(O)
pict = "help.pcx"
retcode = pcxFD(pict,30,80,1)
vfree = pcxVE( pcxCMI*!)
set color to w+/n,w+/n
€10,40 say "WAIT"
€1,1 say " "
header = SPACE(48)
vptr = SPACE(4)
vi         = SPACE(4)
pal = SPACEC17)
        * Calculate the memory required
        vreq = pcxVZ(pcxtype,640,350)
        IF (vreq < vfree)

          *   Yes,   so   create   the   virtual   buffer
retcode = pcxCV(pcxCMM,vptr,pcxEGA_10,640,350)
```

```
          IF (retcode = pcxSUCCESS)

            ♦ Load the image into the virtual buffer
            retcode = pcxFV(name,vptr)

endif
endif

set color to w+/n,w+/n
01,1 say "    "

retcode = pcxVD(vptr,0,0,10,50,630,340,0)
kart =1
01,1 say " "
DO rama

DO menu

settext()
pcxSM(pcxtext)
t*settext()
set cursor off
DO zvuk

return


Ж * * * * * * * * * *ЖЖ * Ж Ж Ж Ж)((ijcЖЖЖ* Ж ЖЖЖ ifЖ* Ж * Ж)((Ж Ж if Ж
ЖЖ ФУНКЦИЯ ПРОВЕРКИ НАЛИЧИЯ ДРАЙВЕРА MOUSE *Ж
ЖЖЖЖ**ЖЖ           RESMOUSE            ЖЖЖЖЖЖЖЖЖЖЖ

nbuttons = mreset()                    && установка mouse

.if nbuttons = o                       && есги mouse присутствует ?
@11,28 TO 15,55 DOUBLE
wshadow (11,28,15,55)
   @ 12,29 say 'И Ы 111 А не инсталированна,'
0 13,29 say ' Придется ВЫЙТИ В DOS и '
Є 14,29 say ' исправить погрешность,
do while inkey() =0
enddo
return
endif

return

    a^ a^     dl ala ala a^ a^ ^a ^a ^a   ala a^ a^ a^ Ца a^ ala    a^ a^ ^a

              a^
^k ^k ^k ^k ^k ^k ^k T' ^k ^k ^k ^k ^k ^k ^k ^k ^P ^k ^k ^P ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^P ^P ^k
ЖЖЖ           ПРОЦЕДУРА ВХОДА В РЕДАКТОР,                  ***

ЖЖЖ УСТАНОВКА РЕЖИМОВ,СОЗДАНИЕ ВИРТУАЛЬНОГО БУФЕРА **Ж
ЖЖЖЖЖЖЖЖЖЖЖЖЖЖ         START.PRG          ЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖ
                ф a^           ab a^    ab
^k ^k ^к ^к ^к ^к ^к    ^к Ж  ^к *P V ^к ^к ^к Ф ^к ^к     ^к ^к    Ф ^к
```

```
   retcode = pcxSM(pcxGRAPHICS) && Установка режима графики
   sethires(0)
   pict = "help.pcx"
   retcode = pcxFD(pict,30,80,1)        && загрузка файла HELP
   vptr = SPACE(4)
   retcode = pcxCV(pcxCMM,vptr,pcxEGA_io,640,350) && создание
                                                 Л& виртуального буфера
   new = 0
   DO rama
   DO menu
   settext()
   pcxSM(pcxtext)                     && Установка текстового режима
   set cursor off
   DO zvuk
   return

 if» ifc ifc ifc ifc ifc ifc ifc ifc ifc ifc 1ft ifc ifc ifc ifc ifc ifc ifc i|c ifc
ifc ifc ifc )f( ifc ifc ifc ifc ifc ifc ifc ifc ifc * ifc ifc ifc ifc ifc ifc ifc ifc
                    ifc ifc ifc ifc ifc ifc ifc ifc ifc ifc ifc ifc
        **                      МЕНЮ РЕДАКТОРА                                  **

   level2=8
   DO WHILE.NOT.Ievel2=7
   SET COLOR TO G+/n,rb+/n
   boxfill (170,893,900,80,0,cyan)

   @1,12 PROMPT 'СИМВОЛ'
   @1,21 PROMPT 'ЛИНИЯ'
   01,29 PROMPT 'КРУГ'
   @1,34 prompt 'СТИРАТЬ'
   @1,42 PROMPT 'HELP'
   01,49 PROMPT 'ЗАПИСЬ'
   01,57 PROMPT 'ВЫХОД'

   MENU TO level2
   DO CASE
            CASE level2=l
   clrwin(170,893,1070,973)
   DO prosk
   CASE level2=2
   DO LINE
   set color to g+/n,gr+/n
            CASE level2=3
   DO circle
   CASE level2=4
   DO stir
            CASE level2=5
   pcxSP(l)
   inkey(O)
   pcxSP(O)
   CASE level2=6
   DO sv
   CASE level2=7
   SET COLOR TO W/N+,GR+/RB
   retcode = pcxDV(vptr)
```

oppua
aseopua
uunpu
(раи^тііzомха'оббЧЕЕТ^^итхоч
(aii4M‘TIIJONXa‘OOOT‘OS£T‘O‘O)II J±xoq
(000T‘8bT‘08Z‘OT)ᵁTᴹᴶI³
saseqrq.?p asojo
yye asop
д=£іалаі 3SV3
asn
as?qe|Fp asojo
I>j щім soud 00
qooudoaA asn
ST = I>1
Ь=£іалаі 3SV3
asn
aseqp^ep asoja
1>i ч;тм soud 00
1UOJ.33A asn
T Toaias
иоениз T PH i^аоииеє оэхзанш/ох $=s g=n
£=£іала]- 35У3
asn
as?q?}Fp aso{3
T>1 НИИ soud 00
}.ал5эал asn
0Z = n
г=£іалаі 3SV3
asn
aspq?-p?p asoja
I>| qpw soud 00
уаиіззал asn
ST = W
T=£іалаі 3SV3
3SV3 00
Еіалаї oi ON3W
,и0ХЯа. IdWOdd T‘ьй
.HBhOdU. IdWOdd T‘£S
.juM/idm. idwoad T'Z&
.•сьоізаз. idwodd T'T©
,HXJ3cLL3. IdWOdd T‘00
u/+u6‘u/+g oi У0Ю3 13S
S=£Tanai’iON'3HHM.OO
9=£іалаі
ф ф ф ф «ь ьЬ ф ф si¹
x}*
ж**»** bud">|soud
****** VdinOJdLJ 0H3W зонаоюо **
^|c)|(^|C ї|C)|( )|(^|c j|c 5|c)|(^|c)|(
uun|au
oppua
aseDpua
j.j.o uosuna ^as
II[e aso{3

```
***********                    ПРОСМОТР                    ************
***** Перед входом необходимо открыть базу и передать *********
***** количество записей на один элемент                   *********
                              14 X       1/1               п* П' "г* п* " т' " Ф £££££££££
*****                    PROS.PRG                          *********
 з(с * * * * * * * * * * )* * * * * * * * * *)((* )|( )|( * * * * jjc * * * * *

* ijc * <* )|( * * * )(c *)((* *)((* * * * * * * * * * * *

set escape on

DO WHILE .NOT. E0F()                    && Выводить до конца базы


set color to b+/n,r+/n
01,9 say ""
I = 0
fed = 10
DO WHILE I<fed                          && ВЫВОДИТЬ ПО 10 записей
   if                   i                  =                  5
   02,9                          say                           ""
   endif
   0row(),col()+l                  PROMPT                    char
   i                 =           i            +               1
   skip kl                     && количество записей на 1 элемент
enddo
asa = recnoO                       && Запомнить положение указателя
0row()+i,ю PROMPT 'дальше'
0row(),17 prompt 'Вперед'
0row(),24 PROMPT 'ПОИСК'
set color to b+/n,r+/n
MENU TO com
skip -(fed*kl)                          && установка указателя в начало

if com = 0
use
close all
set color to w/n,gr+/n
clrwin (155,820,1100,990)
set escape off
return
ENDIF
if com < fed+1                            &&.Показ элемента
                  skip (com*kl)-kl

                   vel = 0.0

  0 3,31 say 'Введите увеличение элемента 0-2:'get vel picture "9.9"
set cursor on
read
set cursor off
SET COLOR TO W+/N,W+/N
```

```
                else
        if com=fed+2
        skip -((com*kl)-(2*kl))
        else

               if com = fed+з                    && Поиск элемента
                set color to gr+/n,r+/n
                     a = 0
                do while a=0

                     aa = space(ll)

                     63,31 say 'Введите элемент поиска:-' get aa
      set cursor on
      read
      set cursor off
      if aa=“ ”
            63,31 say 'Нет элемента.Продолжать?'
      do otvetl
      if otvet =1
      set color to gr+/b+
      63,31 say '
      loop
      else
      set color to gr+/b+
      63,31 say '
      a = 1
      endif
      else
      aa= rtrim(aa)
      locate for char = aa
      if .not. found0
      63,31 say '
                          63,31 say-"Элемент не найден.Продолжить?"
            do otvetl
            if otvet = 1
            set color to gr+/b+
            63,31 say '
            loop
            else
            go asa
            skip -(fedfckl)
            set color to gr+/b+
            63,31 say '
            a = 1
            endif
            else
            a = 1
            endif
            endif
            enddo
            release scr           -
    else
go asa
endif
```

```
endif
endif
ENDDO
use
close all
clrwin (155,820,1100,990)
set escape off
RETURN

     к xl»      ^xk xl/ ^k ^k xk xk ^k ^k ^k ^k ^k 'k ^k *k xk ^k ^k ^k ^k ^k ^k ^k ^k xk ^k ^k ^k xk xk ^k ^k xk ^k ^k xk ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k xk
** ПРОЦЕДУРА ОБСЛУЖИВАНИЯ ИЗОБРАЖЕНИЯ ЭЛЕМЕНТОВ **
«*                       БАЗ ДАННЫХ                              **
»#                     . MOUSE.PRG                               »*
 кккккккккккккккккккккккккккккккккккккккккккккккккк

private г, nbuttons, button, icode, ctype, oldctype, x, у
private xl, yl, clr
private xold,yold,START
START =0
wait = 1
ris = 2
del = 1
LHODE = 2
clrwin(150,815,1350,1000)
if new = 1
if kart = 1
set color to w+/n,w+/n
«1,1 say " "
retcode = pcxVD(vptr,0,0,10,50,630,340,0)
retcode = pcxDV(vptr)
kart =0
else
set color to w+/n,w+/n
.«1,1 say" "
retcode = pcxPI(vptr,0,10,50,0)
retcode = pcxDV(vptr)
endif
елгi i f
saystring(1180,910,0,0,WHITE,'X = ')
saystring(1180,870,0,0,WHITE, 'Y = ')
boxfill(0,0,1350,1000,BXNOFILL,white)
boxfi 11(6,6,1335,990,BXNOFILL,red)
r   =msetwin(o,o,1350,750)           &&  ограничение движения курсора
г   =                  mfixpos(700,450) &&  инициализация стартовой позиции
г   =                          mcuron() &&  включение курсора
ctype = 0
xl=mgetx()
x2=mgetx()
yl=mgety()
y2=mgety()
do while inkey() 0-9
   button = mstatusO             -    && КОЛ-ВО КНОПОК
   do case
   case button =1                     && левая кнопка
if start =0
```

```
if wait = 1
         xl = mgetx()
         yl = mgetyO
         Imode = 2
         select 2
              do vf with xl,yl,0,0,0,Imode,15,0,kl
select 1
ris = 1
start = 1
wait = 0
del = 1
else
ctype = 0
start = 0
wait = 1
endif
endif
  case button =2                      && правая клавиша
start = 0
if del = 1
Imode = 1
select 2
              do vf with xl,yl,0,0,0,Imode,15,0,kl
select 1
ctype = 0
ris = 0
endif
del = 0
wait = 1
endcase
if ris = 1
  if mmotionO <> 0                  && ВЫЛО ЛИ ДВИЖЄНИЄ ?
   ~r = mstatusO                    && регистрация текущей позиции

select 2
  do vf with xl,yl,0,0,0,Imode,15,0,kl
select 1

    x1 = mgetx()
yl = mgety()••
saystring (1230,910,0,0,LIGHT+BLUE,.str (xl, 5))
saystring(1230,870,0,O,LIGHT+BLUE,str(yl,5))
CTYPE =1
select 2
do vf with xl,yl,0,0,0,Imode,15,0,kl
select 1
endif
endif                              ■
Imode = 2
r = mcurtype(ctype)
enddo
```

```
жж----------------------------------------------------------------------------------------------------- - ----------------------------------------------------------
г = mcuroffO                          && вьключение курсора
clrwin(1175,855,1335,975)
drawline(0,0,0,1000,0,0,white)
drawline(6,6,0,990,0,0,red)
drawline(O,O,O,0,1000,O,white)
close database
use
close all
SET COLOR TO w/N,W/N
retcode = pcxCV(pcxCHГI, vptr,pcxEGA_10,640,350)
retcode = pcxGI(vptr,10,50,630,250,0)
kart =0
new = 1
do гала
set cursor off
return


ЖЖЖЖ               ПРОЦЕДУРА РИСОВАНИЯ ЛИНИИ                    ЖЖЖ
ЖЖ                          LINE.PRG                            ЖЖЖ
*
private г, nbuttons, button, icode, ctype, oldctype, x, y
private xl, yl, clr
private xold,yold,START
START = 0
wait = 1
ris = 2
del = 1
clrwin(150,815,1350,1000)
if new =1
if kart = 1
set color to w+/n,w+/n
ei,l say " "
retcode = pcxVD(vptr,0,0,10,50,630,340,0)
retcode = pcxDV(vptr)
kart =0
else
set color to w+/n,w+/n
01,1 say " "
retcode = pcxPI(vptr,0,10,50,0)
retcode = pcxDV(vptr)

                 endif
endif
saystring(1180,910,0,0,WHITE,'X = ')
saystring(1180,870,0,0,WHITE, 'Y = ')
boxfill(0,0,1350,1000,BXNOFILL,white)
boxfill(6,6,1335,990,BXNOFILL,RED)
r = msetwin(8,10,1340,750) && определение окна для движения
&& курсора
г = mfixpos(700,450)            && инициализация начальной позиции
г = (neuron ()                  &&. включение курсора
```

```
xl=mgetx()
X2=mgetx()
Yl=mgety()
Y2=mgety()

ctype =0

do while inkeyO 0-9          && ВЫХОД через F-10

   button = mstatusO && определение нажатой клавиши
   do case
   care button = 1           && левая клавиша

     if wait = 1
          Xl = mgetx()
          Yl = «igety ()
          X2 = mgetx()
          Y2 = mgetyO
       rir = 1
       start = 1
       wait = 0
       del = 1
       else

           ■> if start = 1

                  Xl - mgetx()
                  Yl = mgety()
drawline(x2,y2,xl,yl,0,0,15)
                  X2 = mgetx()
                  Y2 = mgetyO
                  ctype = 1

             endif

endif
   care button = 2                •     && правая клавиша
if del =1
                drawline(x2,y2,xl,yl,2,0,15)
              ctype = 0      '•
              ris = 0
endif

del = 0
wait =1

   endcase
if ris =1
   if mmotionO <> 0               && ВЫЛО ЛИ ДВИЖЄНИЄ?
      r = mstatus()               &&


       drawline(x2,y2,xl,yl,2,0,15)

     xl = mgetx()
```

```
     yl = mgety()
     saystring(1230,910,0,0,LIGHT+BLUE,str(x1,5))
saystring(1230,870,0,0,LIGHT+BLUE,str(yl,5))
ctype = 1
drawline(x2,y2,xl,yl,2,0,15)
endif
endif

     r = mcurtype(ctype)
enddo

r = mcuroffO                && выключение курсора
clrwin(1175,855,1335,975)
boxfill(0,0,1350,1000,BXNOFILL,WHITE)
boxfill(6,6,1335,990,BXNOFILL,RED)
retcode = pcxCV(pcxCMM,vptr,pcxEGA_10,640,350)
retcode = pcxGI(vptr,10,50,630,250,0)
kart = 0
new = 1
do rama                                                        ■>
set cursor off
return

               Mr          ФФФ                & Mr Ф Ф Ф          Mr Ф Mr tV Mr & Ф ^k ^k ^k
                              ^k ^k * ^k ^k Ф ^k ^k ^k ^k          ^k ^k ^k ^k ^k ^k * ^k ^k ^k ^k
**             ПРОЦЕДУРА РИСОВАНИЯ ОКРУЖНОСТИ                  ***
**                         CIRCLE.PRG                           ***
^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k r^ ^k ^k ^k ^k ^k ^k ^k ^k ^k
private r, nbuttons, button, icode, ctype, oldctype, x, y
private xl, yl, clr
private xold,yold,START
START = 0
.wait = 1
ris = 2
del = 1

clrwin(150,815,1350,1000)
if new = 1
                 if kart = 1
set color to W+/n,W+,n
€1,1 say " "
retcode = pcxVD(vptr,0,0,10,50,630,340,0)
retcode = pcxDV(vptr)
kart = 0
else
set color to w+/n,w+/n
€1,1 say " "
retcode = pcxPI(vptr,0,10,50,0)
retcode = pcxDV(vptr)

                 endif
endif
saystring(1180,910,0,0,WHITE,'X = ')
saystring(1180,870,0,0,WHITE, 'Y = ')
```

```
boxfill(0,0,1350,1000,BXNOFILL.white)
boxfill(6,6,1335,990,BXNOFILL,RED)
r = msetwin(8,io,1340,750) && ограничение движения курсора
r = mfixpos(700,450)           && инициализация стартовой позиции
r = mcuronO                    && включение курсора
x1 = mgetxO
x2 = mgetx()
yl = mgety()
y2 = mgety()

ctype = 0

do while inkeyO 0-9           && ВЫХОД через F-10

   button = mstatusO           && определение нажатой клавиши
   do care
   case button = 1             && левая клавиша

    if wait = 1
         XI = mgetx()
      Yl = mgetyO
      X2 = mgetx()
      Y2 = mgetyO
      ГI5 = 1
      start = 1
      wait =0
      del = 1
      «ode = 1
      else

            if start » 1

                  XI = mgetx()
                  Yl = mgety()
drawcircle(xl,y1,abs(x2-xl),0,360,0,0,15)
                  X2 = mgetxO
                  Y2 = mgety()
                  MODE = 0
            ctype = 1
            wait =0
            endif

endif
   case button =2                        && правая клавиша
if del = 1
             drawcircle(xl,yl,ABS(x2-xl),0,360,2,0,15)
             ctype =0
             ris = 0
endif

del = 0
wait =1

   endcase
```

. . Xes т‘10
N/+«‘N/+M OI aoюз 13S
T = juex и

j = паи j.1

(000T‘0S£T‘SI8‘0SI)ujMJn
I = I«P
Z = SJ-*
I = IT?"
o = lavis
u/u‘u/u o-j. uojoo jas

18V1S[1]PJoX‘pjox a|₽ATud
J{3 ‘jX ‘jx a|PATjd
X ‘x ‘adXjDpjo .‘adXjo ‘apoэт ‘uo^nq ‘suojjnqu ‘j ajPATud
«

4 4 ^k 4 ^k * 4 ^k * ™ Φ ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k W ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k ^k Φ v ø ® ® ® ®ø "П *n *F *n
** Odd'HIlS »»
** L5l/HWc4/113 Wd^tQTlOdU *t

uun;au
±j.o uosjno |as
ew?u op
I = маи
0 = 1J*4

(0‘0ez‘0£:9‘0S‘0T‘ J^dA)igxod = apoэ^аи
(0S£‘0t79‘0rV93xod‘jydA‘wu3XDd)A3X3d = apoзцаи
(0‘8ZT+b9+0Z‘066‘S££T‘9‘9)TIT1[X0]Я
(0‘8ZI+b9+0Z‘000l‘0S£l‘0‘0)TTyj-xoq
(SZ6‘S££1‘SS8‘SZTT)UTMJIO
edcad/vx anMShovxiqa ws ()j.j.ounow = u

oppua
(adX^o)adXq.jnom = j

Ijpua
±xpua

(ST‘0‘Z‘09£‘0‘(TX-ZX)Sav‘lX‘TX)ai3UX3M®jp
I = adX^.3
((S‘lX)-iis‘3nia+lH9I3‘0‘0‘0Z8‘0£ZT)&ui:u|sXes
((S‘lx)Jls‘3ma+lH9I3‘0‘0‘0I6‘0£Zl)buTJ;sX?s

()X|абш = jX
Ox^абш = jx

(Sl‘0‘Z‘09£‘0‘ (Ix-zx)S8W‘T><‘Tx)aJ3 j jDMeup

ииїїиєои ii?aW\>ei bMhedxoi/uad «« ()snj?jsui = J

саинажийС ио алча w o <> ()UOIJ.OWW ±т
т _ [5]т[J] л

```
retcode = pcxVD(vptr,0,0,10,50,630,340,0)
retcode = pcxDV(vptr)
kart = 0
else
set color to w+/n,w+/n
01,1 say " "
retcode = pcxPI(vptr,0,10,50,0)
retcode = pcxDV(vptr)

                endif
endif
saystring(1180,910,0,0,WHITE,'X = ')
saystring(1180,870,0,0,WHITE, Y = ')
boxfill(0,0,1350,1000,BXNOFILL,white)
boxfill(6,6,1335,990,BXNOFILL,RED)
r = msetwin(8,10,1340,7      )   && Ограничение движения курсора
r = mfixpos(700,450)             && Инициализация стартовой позиции
r =. mcuron ()                   && Включение курсора
xl=mgetx()
X2=mgetx()
Yl=mgety()
Y2=mgety()
ctype = 0
set color to п/п,п/п
61,1 say ' '

do while inkeyO 0-9          && Выход через F-10
button = mstatus()
                             && определение количества кнопок
do case
case button = 1              && Левая клавиша
if wait = 1
         XI = mgetx()
       Yl = mgety()
       X2 = mgetx()
       Y2 = mgetyO
       ris = 1
       start = 1
       wait = 0
       del = 1
       else
       if start = 1

                  XI  =  mgetx(
Yl           =           mgety(
X2           =           mgetx(
Y2           =           mgety(
ctype            =            1
endif
endif
case      button      =      2 && правая клавиша
if        del        =        1
n=0
do while n<20
```

```
drawline(xl,yl+n, xl+20,yl+n,3,0,0)
n=n+l
enddo
               ctype = 0
               ris - 0
endif
del = 0
wait =1
endcase
if ris = 1
   if mmotionO <> 0       && БЫЛО ли движение?
       г = mstatusO && Регистрация текущей позиции
п=0
do while n<20
drawline(xl,yl+n,xl+20,yl+n,3,0,0)
n=n + l
enddo

     xl = mgetx()
yl = mgety()
saystring(1230,910,0,0,LIGHT+BLUE,str(xl,5))
saystring(1230,870,O,O,LIGHT+BLUE,str(y1,5))
ctype =1
endif
endif
     r = mcurtype(ctype)
enddo
r = mcuroffO                            && Выключение курсора
clrwin(1175,855,1335,975)
boxfill(0,0,1350,1000,BXNOFILL,WHITE)
boxfill(6,6,1335,990,BXNOFILL,RED)
retcode = pcxCV(pcxCf1Pl,vptr, pcxEGA_10,640,350)
.retcode = pcxGI(vptr,10,50,630,250,0)
kart = 0
new = 1
do rama
set cursor off
return

*** ПРОЦЕДУРА ВВОДА ИМЕНИ СОХРАНЯЕМОГО ФАЙЛА ***
m                          SV.PRG                            ***

dd = 0
aq = 1
do while aq =1
boxfill(320,400,600,200,0,RED)
boxfi 11(320,400,600,200,BXNOFILL,CYAN)
810,21 SAY "ВВЕДИТЕ ИМЯ СОХРАНЯЕМОГО ФАЙЛА"
вii,30 GET name
set cursor on
read
set cursor off
```

```
name                          =                          upper(name)
if              name              =              ”              "
dd                                =                              1
msgl="       НЕТ       ИМЕНИ       ФАЙЛА.ПОВТОРИТЬ       ?       "
011,32                            say                            “
DO                                                          OTVET1
IF OTVET1 = 2
                  if kart = 1
set color to W+/n,w+/n
01,1 say " "
retcode = pcxVD(vptr,0,0,10,50,630,340,0)
kart = 0
else
set color to w+/n,w+/n
01,1 say " "
retcode = pcxPI(vptr,0,10,50,0)
endif
do rama
return
endif
else
aq=0
endif
if dd = 0
if file(name)=.T.
msgl=' ФАЙЛ '+NAFIE+' СУЩЕСТВУЕТ '•
011,30 say "
011,30 say' ПЕРЕПИСАТЬ?'
DO 0TVET1
if otvetl = 2
                  if kart = 1
set color to W+/n,w+/n
01,1 say " "
.retcode = pcxVD(vptrr0,0,10,50,630,340,0)
kart = 0
else
set color to w+/n,w+/n
01,1 say " "
retcode = pcxPI(vptr,0,10,50,0)
endif
do rama
return
else
aq=0
endif
endif
dd=O
endif
dd=O
enddo
DO save
```

```
*♦**          ПРОЦЕДУРА ЗЯТ/1СИ ФАЙЛА НА ДИСК              ****
*»**                    SAVE.PRG                           ****
  * ^c * з|c з(c * * з|c sjs э|c * * *)(( J(C з(c)((* * * * з|( )jc *
* )jt )jc * sfc ijc э|( * Jjc )|( з(c * з|( jjc )j( jjc jjc jjc
jjc)(()j( # J|C jjc J| ( # * )|c )|(

                  if kart = 1
set color to W+/n,w+/n
€1,1 say " "
retcode = pcxVD(vptr,0,0,10,50,630,340,0)
kart = 0
else
set color to w+/n,w+/n
€1,1 say " "


bufsize = pcxDF(NAME,10,50,630,340,0)
do rama
€2,40 say"0'K"
clrwin(320,400,920,600)
boxfill(280,450,600,150,0,red)
boxfill(280,450,600,150,BXNOFILL,CYAN)
€10,21 say "ФАЙЛ "
€10,C0L()+2 SAY NAME
€10,C0L()+2 SAY "СОХРАНЕН"

inkey(0)
if kart = 1
set color to w+/n,w+/n
€1,1 say " "
retcode = pcxVD(vptr,0,0,10,50,630,340,0)
.kart = 0
else
set color to w+/n,w+/n
€1,1 say " "

retcode = pcxPI(vptr,0,10,50,0)

                  endif
do rama
return


***                      HELP                              ***

€10,20 clear to 20,60
€10,20 to 20,60
set color to g+/bg
€10,36 say " HELP "
set color to r/bg
€12,22 SAY CHR(27)+CHR(24)+CHR(25)+CHR(26)
€12,26 say"- ПЕРЕМЕЩЕНИЕ ПО МЕНО"
€13,22 SAY"ENTER - ВЫБОР ПУНКТА МЕНЮ"
```

```
614,22 SAY'T-10 - ВЫХОД ИЗ РЕЖкПА РЕДАКТИРОВАНИЯ"

INKEY(O)
set color to gr+/b,gr+/rb
610,20 clear to 20,60
SET COLOR TO R/BG,GR+/RB
RETURN


** ФУНКЦИЯ ИЗОБРАЖЕНИЯ ЭЛЕМЕНТОВ БАЗ ДАН-fclX ***
**                          VF.PRG                              ***


 **
 *“       x             && координаты выводимого символа
 **       y
 **       ang
 **       Imode       && Тип ЛИНИИ (OR, XOr, etc.)
 “*       mode             && Режимы рисования линии
 **       model
          Iclr             && Цвет линии
 #*       kl               && количество записей на символ
 **
 “*       do vf  with x,y,ang,mode,model,Imode,Iclr,this,kl
*#

parameter x ,y,ang,mode,model,Imode,Iclr,this,kl
private rad,normwid,width,chnum,baseindx
  r = fixpos(x,y)              && Установка начальной позиции.
  baseindx = this*ki           && кол-во используемых строк базы
  rad = 0
  r = datarange(baseindx,baseindx+(kl-l))
  r = polyvec(rad,ang,1+mode+model,ang,Imode,0,Iclr)
                                                 && Изображение символа
.return


** ГРОЦЕДУРА otvetl                        **


PUBLIC otvetl
otvet=2
610,21 say “
610,21 SAY ‘ '+msgl+' '
*#611,30 say"
612,31 PROMPT ' ДА
612,38 PROMPT 'HET '
MENU TO otvetl
RETURN


**        ПРОЦЕДУРА otvet **


PUBLIC otvet
save screen to s
```

```
otvet=2
SET COLOR TO GR+/R,RB/G,,,W/N
613,22 CLEAR TO 17,56
613,22 TO 17,56 DOUBLE
wshadow (13,22,17,56)
613,24 SAY ' '+msgl+"
617,27 PROMPT ' ДА '
617,50 PROMPT 'HET '
MENU TO otvet
SET COLOR TO W/N+,GR+/RB
613,22 CLEAR TO 17,56
restore screen from s
release s
RETURN

Jc r iЖ )Ж 1Ж ^Ж Ж)Ж)Ж 1Ж зЖ зЖ * зЖ зЖ Ж Ж зЖ ЗЖ зЖ зЖ зЖ зЖ ЗЖ ЗЖ
зЖ )Ж )Ж )Ж зЖ ЗЖ зЖ зЖ зЖ зЖ зЖ зЖ зЖ зЖ зЖ зЖ зЖ зЖ 1Ж зЖ зЖ зЖ зЖ
зЖ )Ж зЖ зЖ зЖ зЖ«зЖ )Ж Ж зЖ зЖ
*!
*1 Function: WSHAD0W(<yl>,<xl>,<y2>,<x2>)
*! Notes: Рисует прозрачную тень справа от заданного окна
ж1, ереработаший для '87 вариант функции SHADOW из Clipper 5
зЖ ж ж Ж Ж ж ж ж ж ж ж ж ж ж ж ж ж зж ж ж ж ж ж ж ж ж ж зж ж ж ж ж ж
ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж ж зж ж зж ж ж ж ж
FUNCTION WShadow
PARAMETERS Y1,X1,Y2,X2
PRIVATE Nyl,Nxl,Ny2,Nx2
Ny1=MIN(Y2+1,24)
Ny2=Nyl+l
Nxl=Xl+l
Nx2=MIN(X2+l,79)
RESTSCREEN( Nyl, Nxl, Ny2, Nx2,;
  TRANSFORM( SAVESCREEN(Nyl, Nxl, Ny2, Nx2),;
  REPLICATECX", Nx2 - Nxl + 1 )+REPLICATE( "XX" ,Nx2-Nxl + l) ) )
Nyl=Yl+l
Ny2=Y2+l
,Nxl=MIN(X2+l,79)
Nx2=Nxl+l
RESTSCREEN( Nyl, Nxl, Ny2, Nx2,;
    TRANSFORM( SAVESCREEN(Nyl, Nxl , Ny2, Nx2),;
    REPLICATE("XXX" , Nx2 - Nxl + 1 ) ) )
```

ПРИЛОЖЕНИЕ 2

ИЛЛЮСТРАЦИЯ РАБОТЫ ПРОГРАММ ПРОЕКТИРОВАНИЯ ОПИСАНИЯ ЖЕЛЕЗНОДОРОЖНЫХ СТАНЦИИ

ИЛЛЮСТРАЦИЯ РАБОТЫ ПРОГРАММЫ ФОРМИРОВАНИЯ И ОБСЛУЖИВАНИЯ БИБЛИОТЕКИ ЭЛЕМЕНТОВ ЖЕЛЕЗНОДОРОЖНЫХ СТАНЦИИ

Основное меню

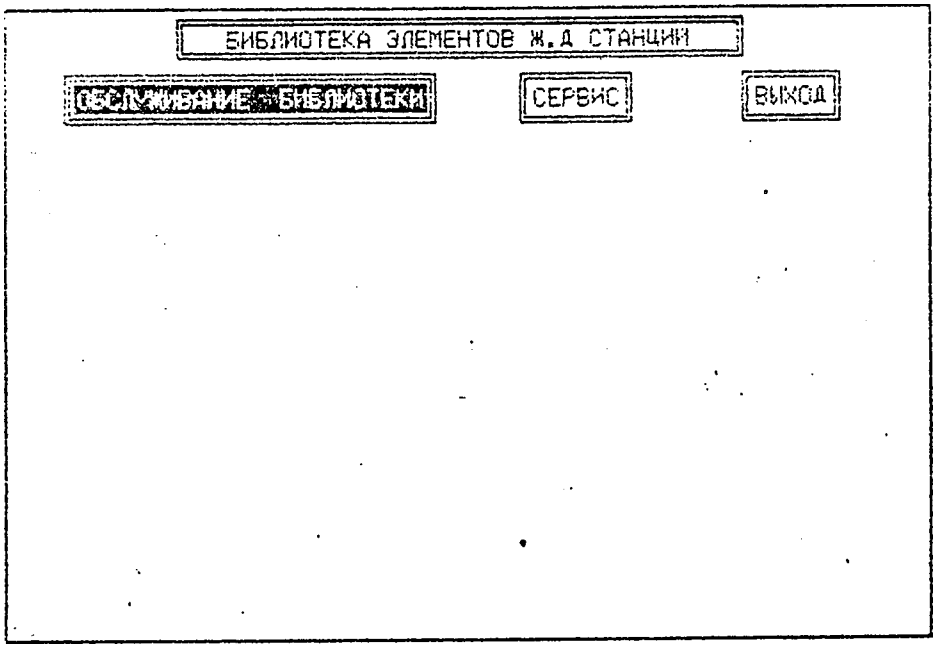


Рис. П.2.1

Меню режима просмотра

Рис. П.2.2

Вид экрана при поиске элемента в режиме просмотра

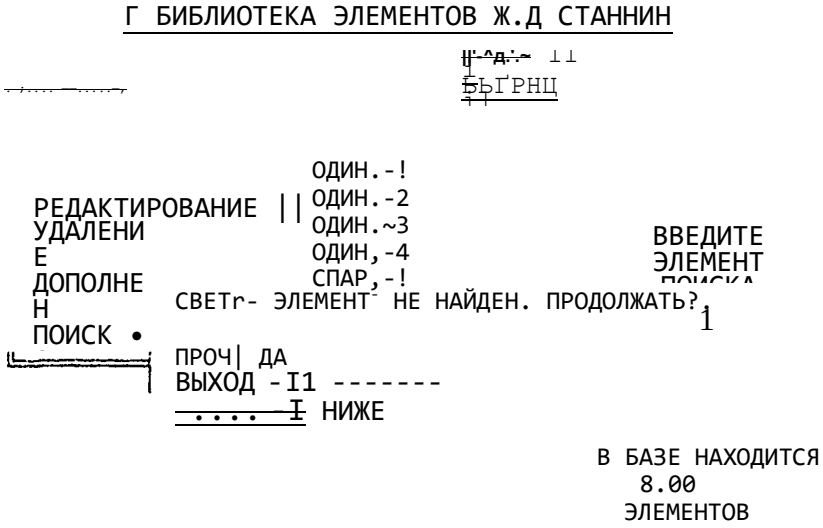


РИС. П.2.3

Элемент "маневровый светофор"

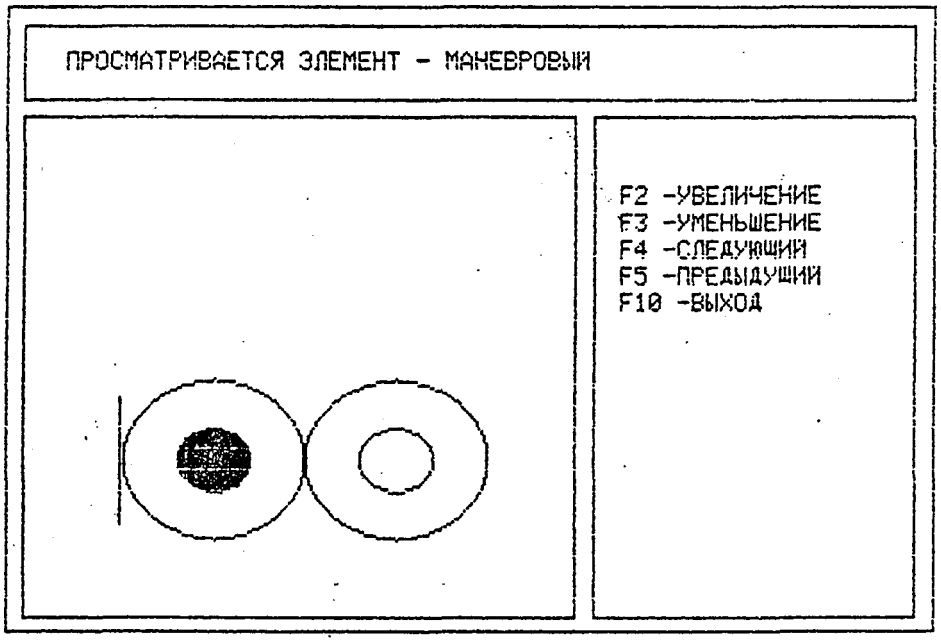


Рис. П.2.4

Меню режима редактирования

j| БИБЛИОТЕКА ЭЛЕМЕНТОВ Ж.Д СТАНЦИЙ ij

ife* пгіт дії^яI СЕРВИС

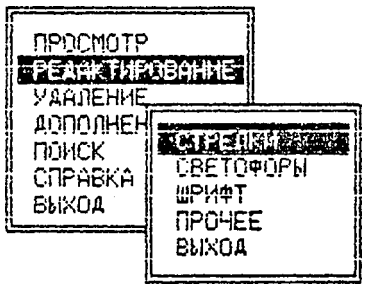
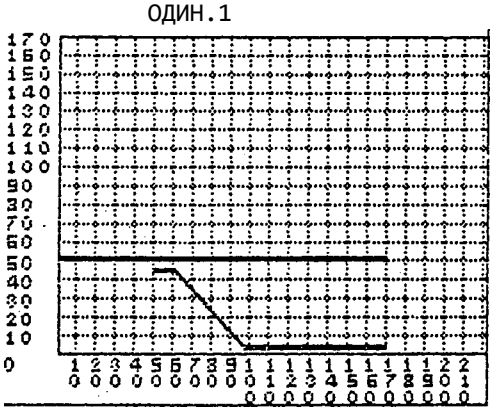


Рис. П.2.5

Формирование элементов

СТРЕЛКИ I

ОДИН.1



[CHAR X Y MOVE

! 8»М!!ЯНЙИ	0	50	1
i r	179	50	0
	53	45	1
	60	45	0
	100	0	0
	170	0	0
	0	0	1
	0	0	- 1
	0	0	- 1
	0	0	1

F1- HELP F2- ПРОСМОТР F3- НАЧАЛО
F4- КОНЕЦ F7- ПОИСК F16- ВЫХОД

-Т I- - ПЕРЕМЕЩЕНИЕ ПО БАЗЕ ENTER- РЕДАКТИРОВАНИЕ

РИС. П.2.6

Вид экрана помощи при формировании элемента

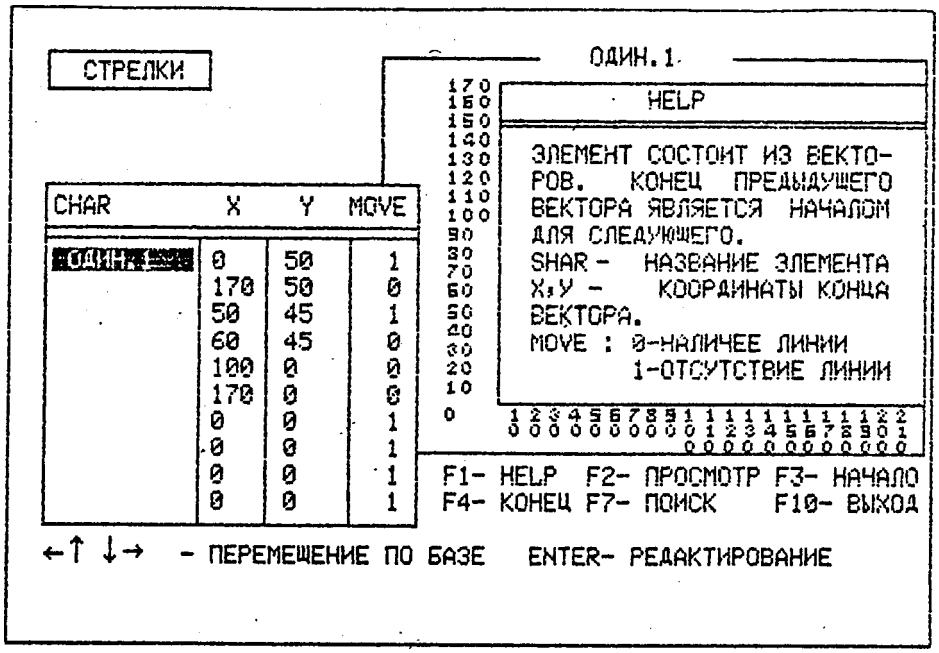


Рис. П.2.7

Меню режима удаления

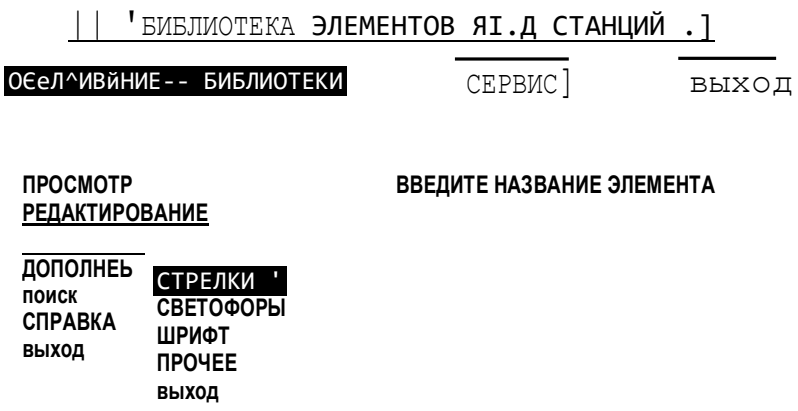


Рис. П.2.8

Удаление элемента

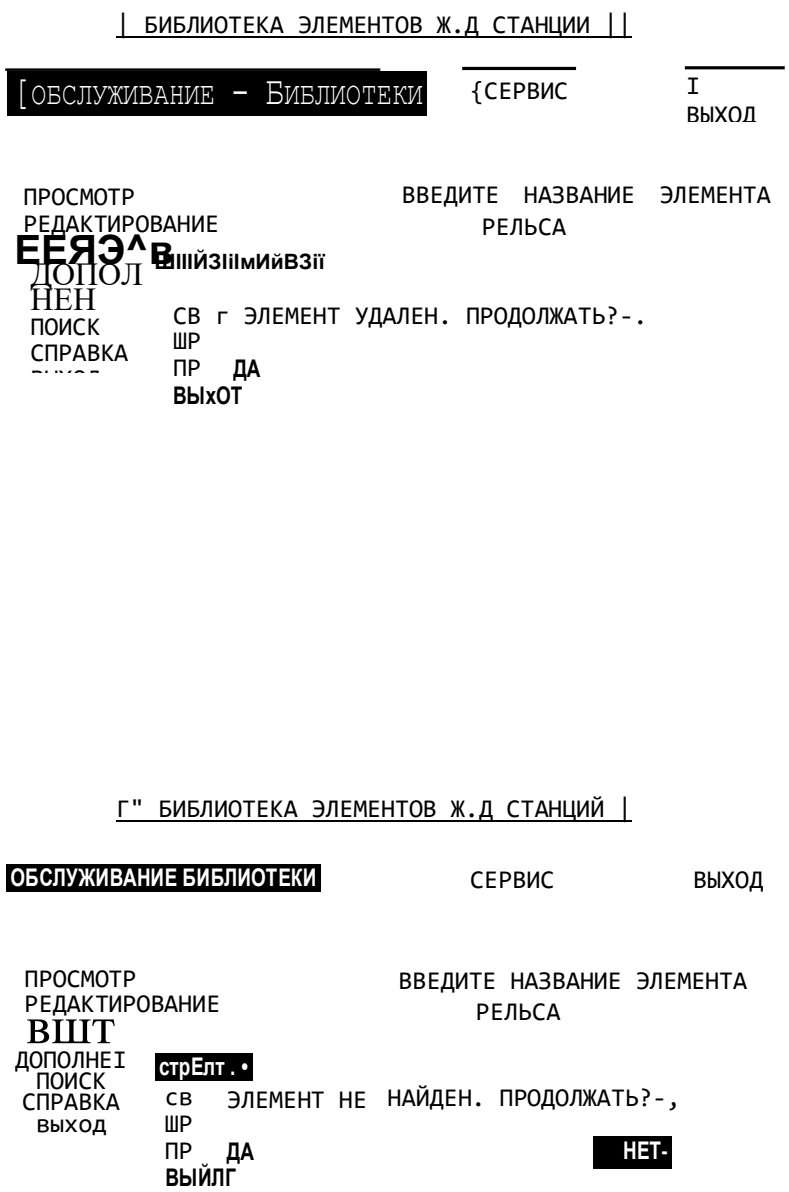


Рис. П.2.9

Меню режима дополнения

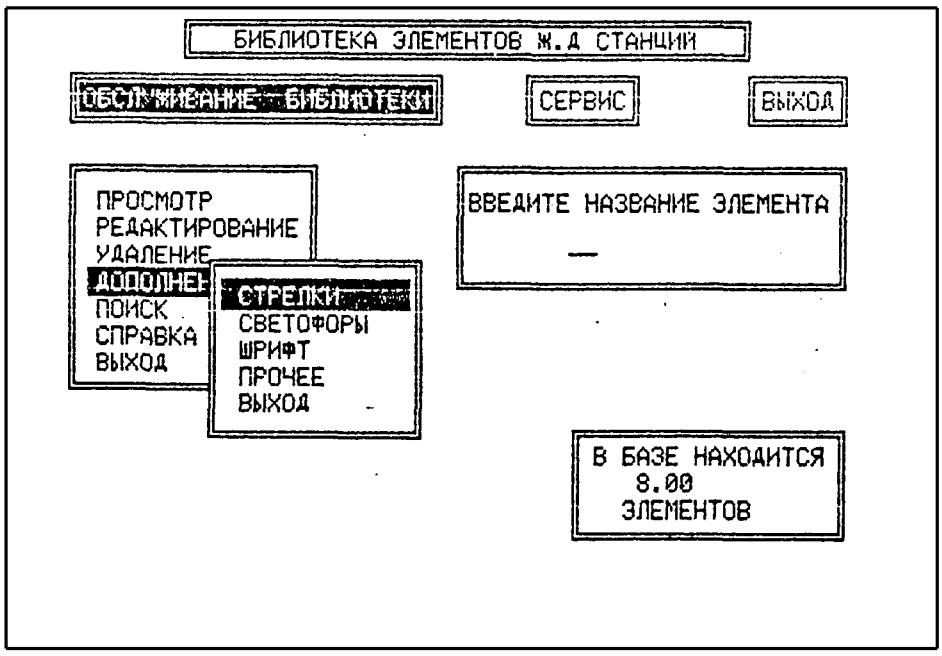


Рис. П.2.10

Запрос на дополнение библиотеки элементов

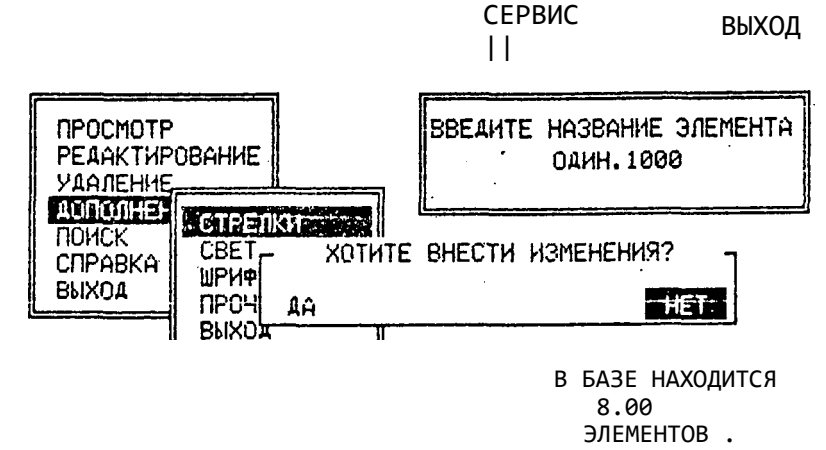


Рис. П.2.11

Дополнение элемента

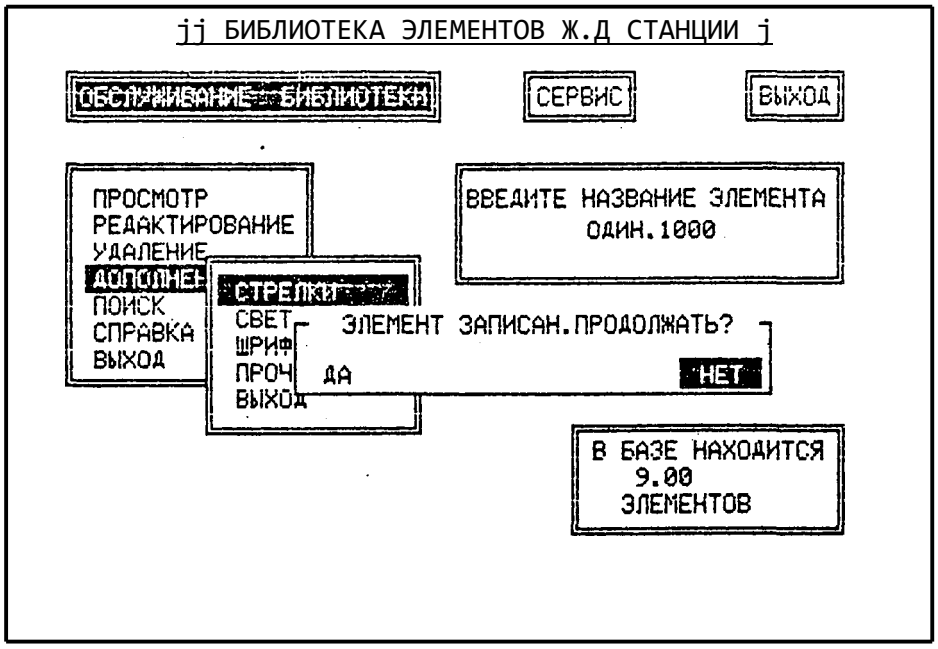


Рис. П.2.12

Меню режима поиска

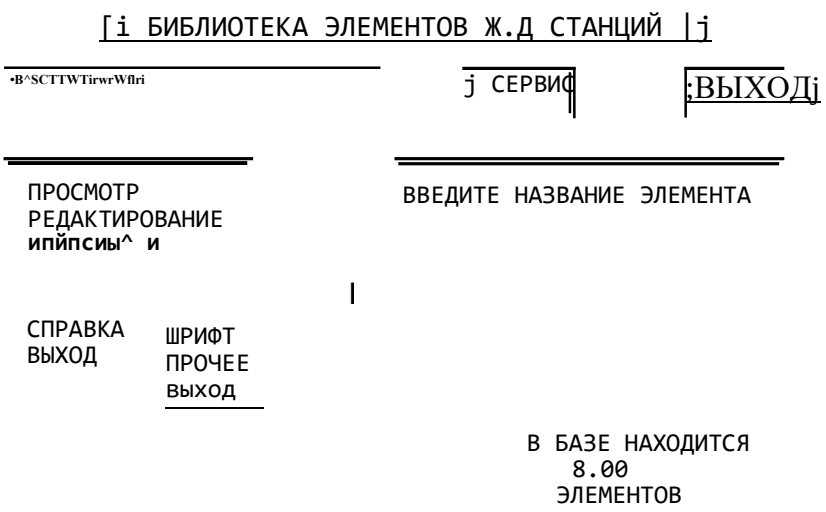
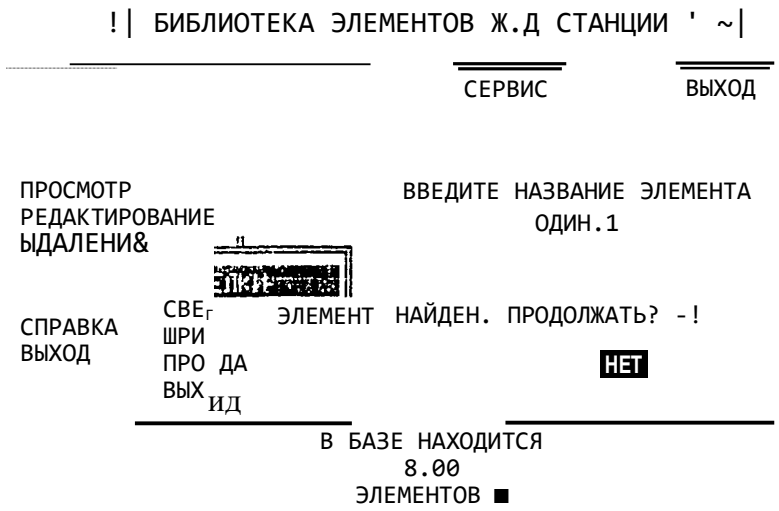


Рис. П.2.13

Поиск элемента



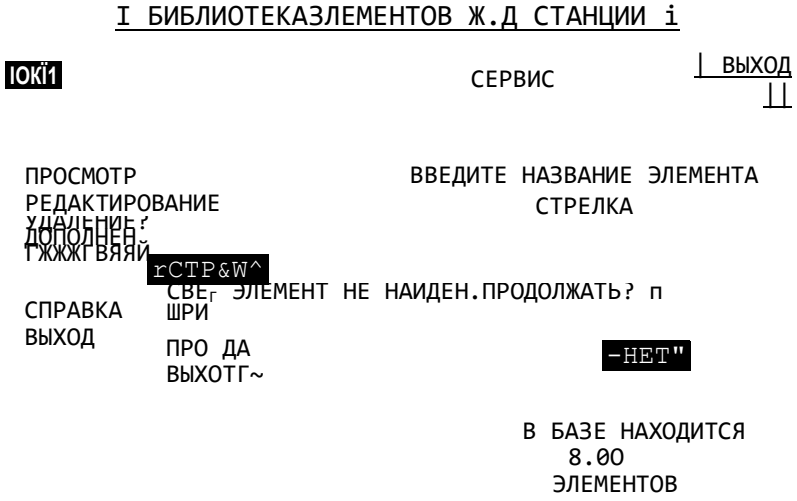


Рис. П.2.14

Меню сервиса

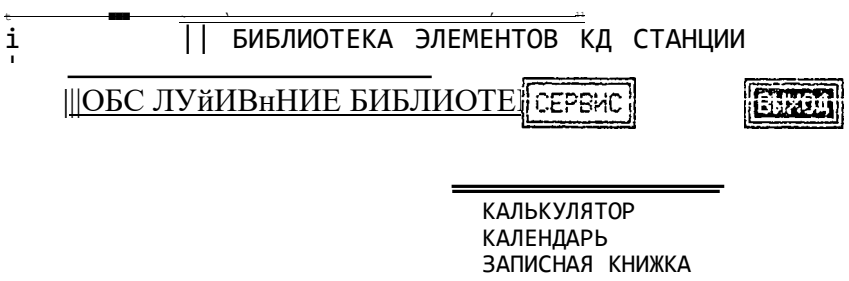


Рис. П.2.15

Выход из программы

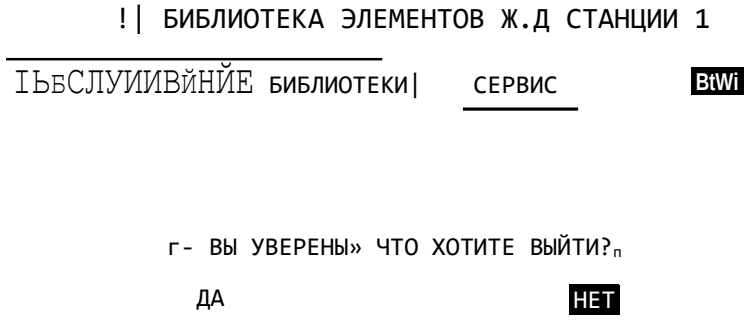


Рис. П.2.16

ИЛЛЮСТРАЦИЯ РАБОТЫ ПРОГРАММЫ ФОРМИРОВАНИЯ СТАТИЧЕСКОЙ МОДЕЛИ ЖЕЛЕЗНОДОРОЖНОЙ СТАНЦИИ

Основное меню

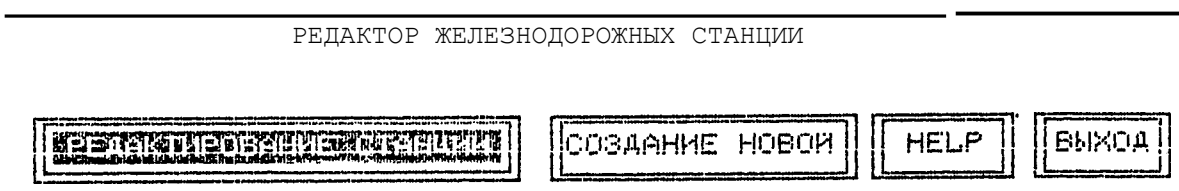


Рис. П.2.17

Меню режима создания схемы новой железнодорожной станции

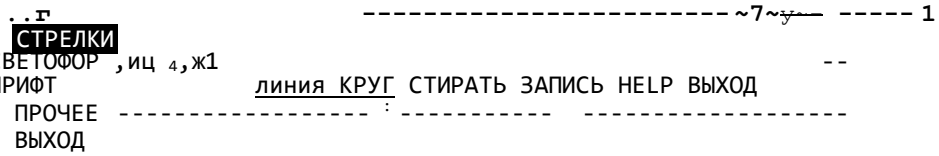


Рис. П.2.18

Вид экрана при поиске элемента в режиме создания

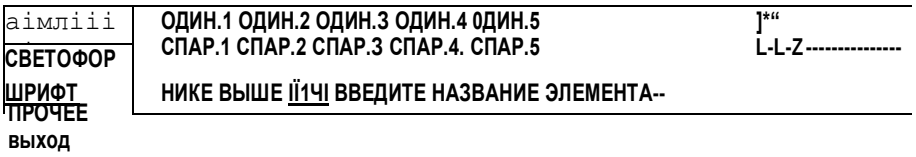


Рис. П.2.19

Вид экрана при использовании директив "символ”,"Линия",’’круг”

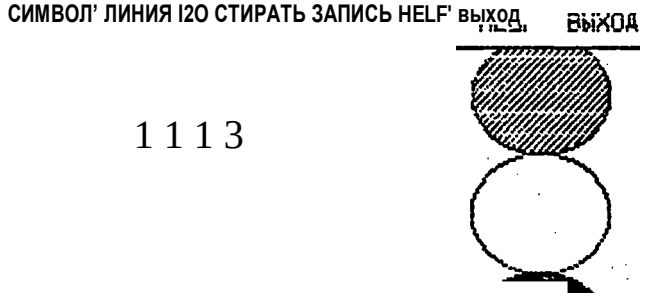


Рис. П.2.20

Запись сформированной схемы станции

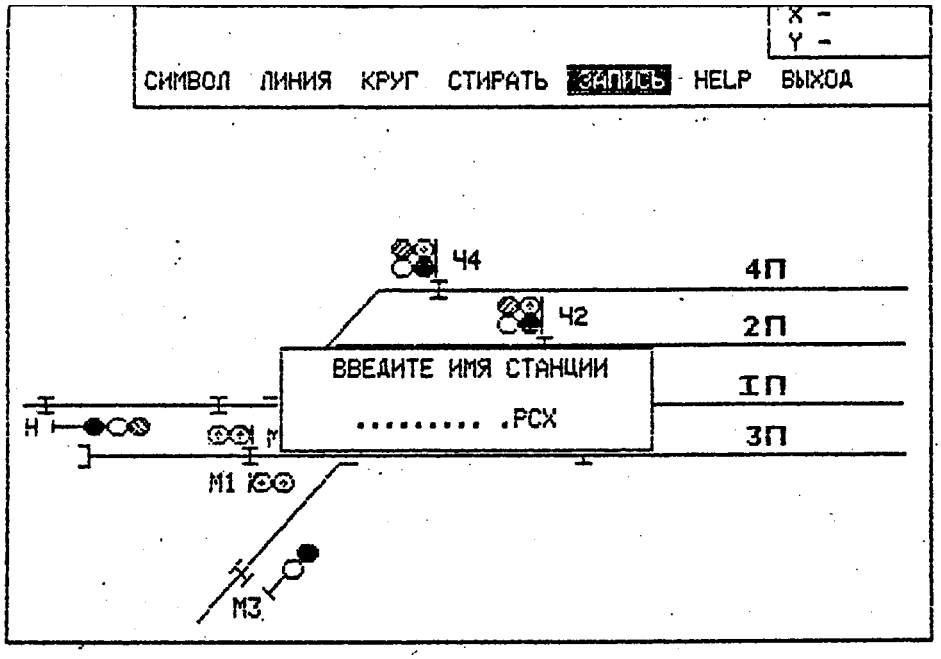


Рис. П.2.21

Вид экрана при входе в режим редактирования

РЕДАКТОР ЖЕЛЕЗНОДОРОЖНЫХ СТАНЦИИ

РЕДАКТИРОВАНИЕ СТАНЦИИ ^СОЗДАНИЕ НОВОЙ HELP ВЫХОД

ВВЕДИТЕ ИМЯ РЕДАКТИРУЕМОЙ СТАНЦИИ
■ . PCX

РИС.

Редактирование схемы станции

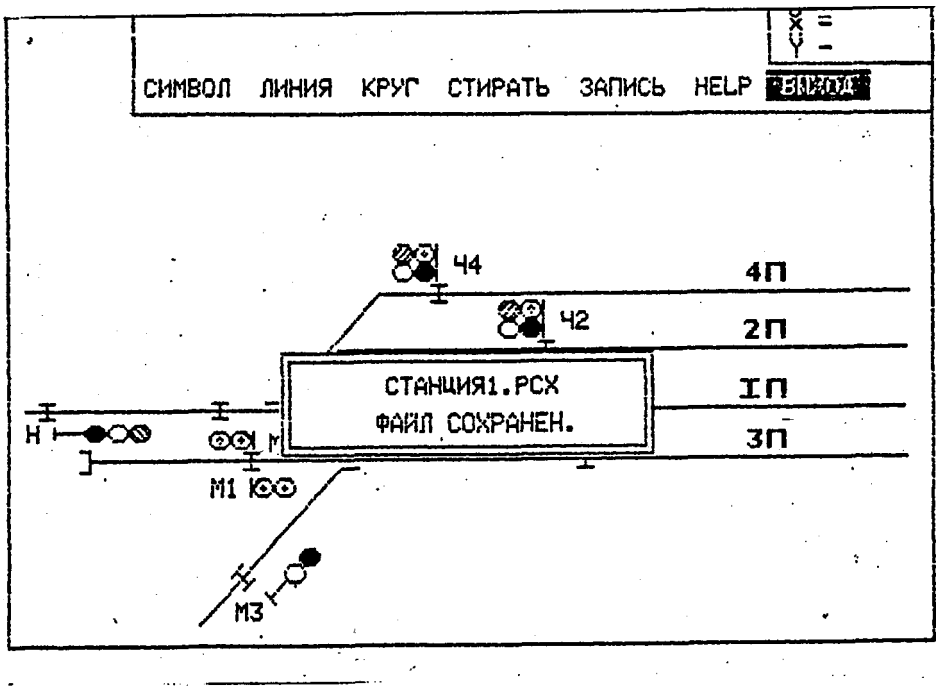


Рис. П.2.23

Выход из программы

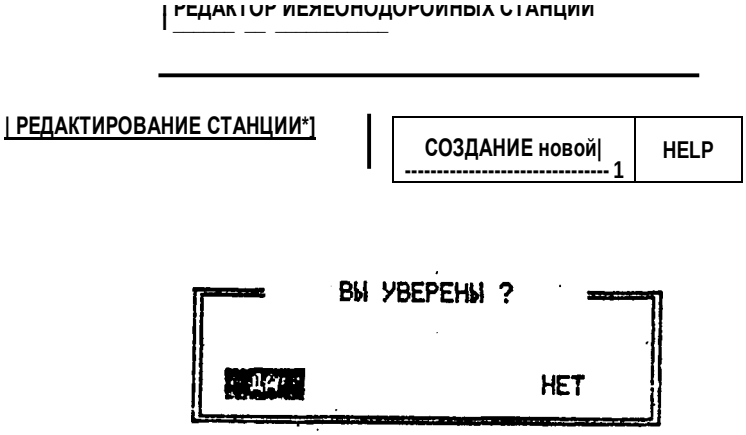


Рис.

ДОКУМЕНТЫ О ВНЕДРЕНИИ И ИСПОЛЬЗОВАНИИ РЕЗУЛЬТАТОВ ДИССЕРТАЦИОННОЙ РАБОТЫ

УТВЕРЖДАЮ:

должность

подпись, ф., и., о.

19 г.

А К Т № * -

20 - декабря 19*36 г.

Харьков

• горож * *

О внедрении разработки, выполненной Харьковским институтом инженеров железнодорож-
ного транспорта им. С. М. Кирова КУ^/Дэт . ? « января 19fU г.

Разработка

сяотемы команд и программного обеспечения для модернизированных устройст

ЛОРлЧеСЛиа ибраиитКЛ .ЛН.^ивЬяйе^Лл/нменование договора

с затратами в размере **- 4С; тыс, руб»** _____ _на _____ _____~ ____

. . тыс. руб. полное неименованно

_____ НПО УМ МЖПРЖОР. _____ ___ •

~_____ предприятия, где внедрена разработка, к его ведомственная подчиненность

Составлен комиссией в составе:

Председатель_____ ПРО у» Кб Д РУС Б. А» _ • ■ ' '

должность, фамилия, и., о.

Члены комиссии: ьаСЛеННИКОВ А.Л.,/~>' _ _____ _____

В период с . I * октябрь 19-JO г. по . 2Гг‘ декабря 19Япг. комиссия провела работу по определению фактического внедрения результатов разработки Уистены

и установила следующее: * ____' • _____ ■ ■ . _____■_ _ _

1. Разработка выполнена в сооответствии с утвержденной рению .‘3** января ___ 19 85г* в сроки •—5L* _____ др.кабря 19 85 г-

фактические сроки внедрения) „~~разработанные предложения по архитектуре У ДОЛ»~~

~~■ входной язык и системный редактор~~

~~редактсфг а~~

■ ’ *f*

) 3. Характеристика результатов внедрения и их оценка. Долевое участие ХИИТа.

В результате внедрения разработок сокращается время проектирования “систем управления на базе УЛО'Л, повышается качество программ контроля и управления. ■

Долевое участие соисполнителе;! следующее:

) доц., к.т.н. Добрянский В.М. - 40'Z, м.н.с. донина С.Jj. = "

м«л«н« Зорина Е.Ц. - 10%, асе. Калиновски:! А.Е. -

4. Экономический эффект,'полученный от внедрения разработки

. составляет 150 тыо.^уб.

ХИЯТ не претендует на вознаграждение за внедрение работ заказчиком.

5. Выводы и предложения 6 дальнейшем использовании разработки

Совершенствование УЛОЛ и сетей на их основе потребует доработки

И соответотвувщего системного резидентного программного обеспечения.

6. Акт исполнен в 5 экземплярах. .

Экз. № 1 ■' НПО’ САУ

Экз. № 2 ХИИТ "

/ . - Экз. № ЗУ 475,’ 6 у" исполнителей •

Председатель комиссии /^4^рКедРУ<> Е*А.„

(>| ***Члены комиссии*** ***(ляу-*** Маоленников А.й.

.. /Прилепа В.Н.

еу^>>ухxX^5рЬрянокий В.М.

Тип. ХИИТа. Ш-1983 г. Зак. 116—3СОО

РОССИЙСКАЯ ФЕДЕРАЦИЯ
АКАДЕМИЯ ЭЛЕКТРОТЕХНИЧЕСКИХ НАУК

111250. Москва, ул. Красноказарменная. 14
Эл. почта: aelsc@srv-m.mpei.ac.ru

Тел.: (095) 273-53-11

_____ №
на №от

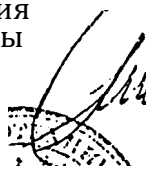
СПРАВКА ОБ ИСПОЛЬЗОВАНИИ

результатов кандидатской диссертации Зориной Е.И. "Технологические аспекты проектирования программного обеспечения систем управления гомогенными объектами (на примере микропроцессорной централизации)"?

Созданная Зориной Е.И. математическая модель функционально однородных (гомогенных) объектов управления позволила создать методику проектирования программного обеспечения (ПО) гомогенными объектами. Использование развитого в диссертации подхода дало возможность создать программные изделия для практических разработок, включая САПР системы управления электромеханическими агрегатами транспортных средств.

Разработанный Зориной Е.И. алгоритм генерации требуемой конфигурации ПО позволил сократить затраты на создание прикладного ПО в НПО "Автоэлектроника'' и Акционерной электротехнической компании "Динамо".

Академик-секретарь
научно-отраслевого отделения
"Электротехнические системы
транспорта и космической
техники АЭН РФ,
д.т.н., профессор



Б.И.Петленко

Г
1'.Л

с

МИНИСТЕРСТВО
ПУТЕЙ СООБЩЕНИЯ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
ПУТЕЙ СООБЩЕНИЯ
(МИИТ)
101475. ГСП-4, Моем»,
ул. Образном, 15
Факе 281-13-40
Телеке 411798 МИТ SU
Телефок

_____

к8о/$Э8

_____ И» № _____

СПРАВКА

об использовании результатов диссертации Зориной Е.И.

"Технологические аспекты проектирования программного обеспечения систем управления гомогенными объектами (на примере микропроцессорной централизации)"

Автор разработал:

1. Новую методику проектирования программного обеспечения (ПО) систем управления (СУ) гомогенными объектами.
2. Схему генерации ПО СУ гомогенными объектами на основе принципов функциональной избирательности и избыточности.

•3. САПР проектировщика описания станции и ПО для автоматизированного создания прикладных программ.

Полученные в диссертации результаты использованы при создании микропроцессорного устройства координатного сближения поездов на метрополитене, при разработке некоторых программ в рамках выполнения хоздоговорных НИР, а также в учебном'процессе.

Проректор универсй/егТа^пб

Oil

У т в е р ж д а ю
проректор ХарГАЖТ по научной
работе, академик, к.т.н.
Жалкин С.Г.
15 марта 1994 г.

ВНЕДРЕНИЯ

результатов диссертационной работы Зориной Е.И.

Результаты, полученные в диссертации Зориной Е.И. на тему: "Технологические аспекты проектирования программного обеспечения систем управления гомогенными объектами (на примере микропроцессороной централизации)", использованы при выполнении хоздоговорной НИР при выборе и обосновании вариантов электрической централизации ЭЦ-Е Сакт-Петербургским государственным проектно-изыскательским институтом "Гипротранссигналсвязь", что позволило автоматизировать структуру ЭЦ-Е для малой станции и обеспечило построение технических средств ЭЦ-Е.

Эффективность результатов диссертации состоит в обеспечении сокращения сроков проектных работ за счет создания универсального программного обеспечения.

Результаты диссертации использованы в учебном процессе.

Зав.кафедрой МИУС,
д.т.н., академик Загарий Г.И.