1. РЕФЕРАТ

В данном дипломном проекте разрабатывается язык процедурного программирования в системе имитационного моделирования РДО (1). Целью данной разработки является внедрение поддержки функционала операций процедурного (императивного) программирования, таких как:

- Поддержка оператор выбора;
- Объявление и инициализация локальных переменных;
- Поддержка циклического оператора;
- Использование областей локальной памяти;
- Поддержка оператора прерывания;
- Поддержка оператора возврата;

В процессе разработки было исследовано быстродействие разных подходов реализации алгоритмов обработки оператора прерывания и оператора возврата. Исследование было проведено на тестовой модели, разработанной в ходе данного дипломного проекта.

В результате был разработан язык процедурного программирования, существенно упрощающий процесс разработки имитационных моделей в системе имитационного моделирования РДО.

1.1. Список иллюстраций

1.2. Перечень ключевых слов

Программное обеспечение, имитационное моделирование, процедурное программирование, оператор, инструкция.

2. ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД – База Данных

В – Внедрение

НИР – Научно-Исследовательская Работа

ПИ – Переменная Информация

ПО – Программное Обеспечение

ПП – Программный Продукт

ППП – Пакет Прикладных Программ

РВ – Реальный масштаб Времени

РП – Рабочий Проект

ТЗ – Техническое Задание

ТПР – Типовое Проектное Решение

ТП – Технический Проект

ЭП – Эскизный Проект

ИМ – Имитационная Модель

СДС – Сложная Дискретная Система

СМО – Система Массового Обслуживания

БЗ – База Знаний

ЭВМ - Электронная Вычислительная Машина

ОЗУ - Оперативное Запоминающее Устройство

3. ВВЕДЕНИЕ

Имитационное моделирование предназначено для прогноза поведения сложных систем, а так же результатов их взаимодействия. Дискретно – событийное моделирование подразумевает такой подход к моделированию, при котором абстрагируются от непрерывности происходящих событий. При дискретно – событийном моделировании считается, что состояние системы меняется мгновенно через сколь угодно короткие промежутки времени. Но в промежутке между двумя ближайшими событиями состояние системы остаётся неизменным.

Язык имитационного моделирования РДО поддерживает дискретно-событийную парадигму имитационного моделирования и реализует три основных подхода моделирования: процессно-ориентированный подход, событийный подход, подход сканирования активностей. Он является проблемно-ориентированным языком, направленным на решение задач моделирования.

Проблемная ориентация обычно производится в контексте некоторого универсального языка программирования, по отношению к которому проблемно-ориентированный язык является либо надъязыком, либо предъязыком, либо подъязыком. Надъязык получается обогащением универсального языка дополнительными конструкциями, особенно удобными для формулирования задачи из класса. В предъязыке дополнительные конструкции полностью "загораживают" универсальный язык и переводятся на него специальным препроцессором. Подъязык получается из универсального языка отказом от конструкций, неупотребительных в данном классе задач, либо предварительным составлением библиотеки "стандартных программ", в совокупности достаточных для выражения любой задачи из класса. Во всех случаях выгода от употребления П.-о. я. состоит в том, что вместо программирования заново каждой задачи из класса достаточно лишь указать средствами П.-о. я. параметры, отличающие одну задачу от другой. РДО является предъязыком в контексте процедурного и объектно-ориентированного языка программирования.

Методика имитационного моделирования часто используется для описания реальных систем. В контексте решения задач моделирования реальных систем часто приходится решать проблемы описания сложных алгоритмических зависимостей. Для описания алгоритмов наиболее удобно использование процедурного (императивного) программирования. Одна из характерных черт процедурного программирования - наличие переменных с операцией "разрушающего присвоения". То есть, была переменная A, было у нее значение X. Алгоритм предписывает на очередном шаге присвоить переменной A значение Y. То значение, которое было у A, будет "навсегда забыто". Для описания синтаксиса будут использованы грамматики в расширенной форме Бекуса-Науэра. В правой части таких грамматик допускается использование следующих "регулярных операций":

- A B - последовательно A, за тем B.
- A | B - альтернатива. Читается: "A или B".
- A* - произвольное количество повторений (в том числе - 0 раз) A. Читается: "последовательность A".
- A # B - эквивалентно A (B A)*. Читается: "последовательность A через B".

Терминальные символы выделяются подчеркиванием.

Синтаксис описания алгоритмов в простейшем языке, поддерживающем процедурную модель программирования, мог бы быть таким:

Оператор ::= Простой оператор | Структурный оператор

Простой оператор ::= Оператор присваивания | Оператор вызова | Оператор возврата

Структурный оператор ::= Оператор последовательного исполнения | Оператор ветвления | Оператор цикла

Оператор присванивания ::= Переменная := Выражение ;

Оператор вызова ::= Имя подпрограммы (Список параметров) ;

Оператор возврата ::= return [Выражение] ;

Оператор последовательного исполнения ::= { Оператор* }

Оператор ветвления ::= if Выражение Оператор* [else Оператор*]

Оператор цикла ::= for (Оператор присваивания Выражение Оператор присваивания) Оператор*

Про реальные системы можно сказать, что они локально императивны. То есть, если взять достаточно узкую задачу, то ее можно вполне легко описать методами процедурного программирования. Процедурное программирование наиболее пригодно для реализации небольших подзадач, где очень важна скорость исполнения.

4. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

4.1. Основные подходы к построению ИМ

Системы имитационного моделирования СДС в зависимости от способов представления процессов, происходящих в моделируемом объекте, могут быть дискретными и непрерывными, пошаговыми и событийными, детерминированными и статистическими, стационарными и нестационарными.

Рассмотрим основные моменты этапа создания ИМ (2). Чтобы описать функционирование СДС надо описать интересующие нас события и действия, после чего создать алфавит, то есть дать каждому из них уникальное имя. Этот алфавит определяется как природой рассматриваемой СДС, так и целями ее анализа. Следовательно, выбор алфавита событий СДС приводит к ее упрощению – не рассматриваются многие ее свойства и действия не представляющие интерес для исследователя.

Событие СДС происходит мгновенно, то есть это некоторое действие с нулевой длительностью. Действие, требующее для своей реализации определенного времени, имеет собственное имя и связано с двумя событиями – начала и окончания. Длительность действия зависит от многих причин, среди которых время его начала, используемые ресурсы СДС, характеристики управления, влияние случайных факторов и т.д. В течение времени протекания действия в СДС могут возникнуть события, приводящие к преждевременному завершению действия. Последовательность действий образует процесс в СДС (Рис. 4.1)

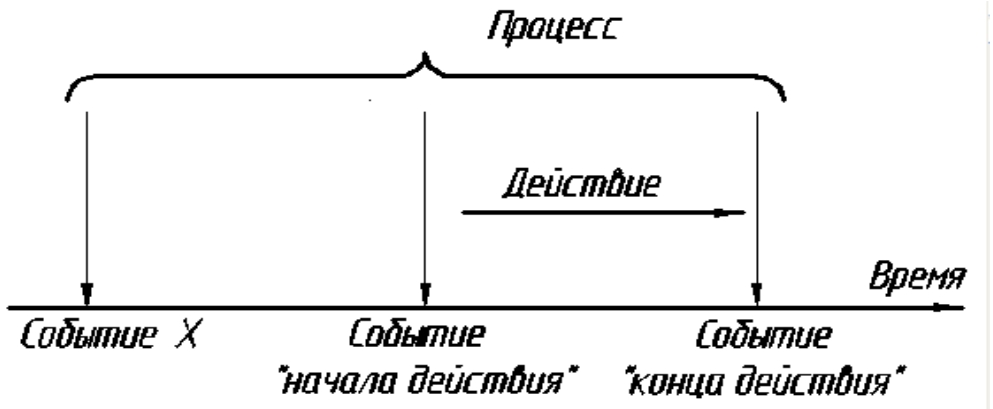

Рис. 4.1 Взаимосвязь между событиями, действием и процессом.

В соответствии с этим выделяют три альтернативных методологических подхода к построению ИМ: событийный, подход сканирования активностей и процессно-ориентированный.

4.2. Процесс имитации в РДО

4.2.1. Процессно-ориентированный подход

Имитационные модели содержат последовательности компонентов, которые возникают в них по определенной схеме, например очередь, в которой клиенты ожидают обслуживания. Логика возникновения компонентов по требуемой схеме может быть обобщена и задана в одном операторе. Имитационный язык затем транслирует такие операторы в последовательность событий, происходящих с компонентами системы.

Процессно-ориентированный подход используется для моделирования систем массового обслуживания (СМО).

4.2.2. Событийный подход

При событийном подходе исследователь описывает события, которые могут изменять состояние системы, и определяет логические взаимосвязи между ними (Рис. 4.2). Начальное состояние устанавливается путем задания значений переменным модели и параметров генераторам случайных чисел. Имитация происходит путем выбора из списка будущих событий ближайшего по времени и его выполнения. Выполнение события приводит к изменению состояния системы и генерации будущих событий, логически связанных с выполняемым. Эти события заносятся в список будущих событий и упорядочиваются в нем по времени наступления. Например, событие начала обработки детали на станке приводит к появлению в списке будущих событий события окончания обработки детали, которое должно наступить в момент времени равный текущему времени плюс время, требуемое на обработку детали на станке. В событийных системах модельное время фиксируется только в моменты изменения состояний.

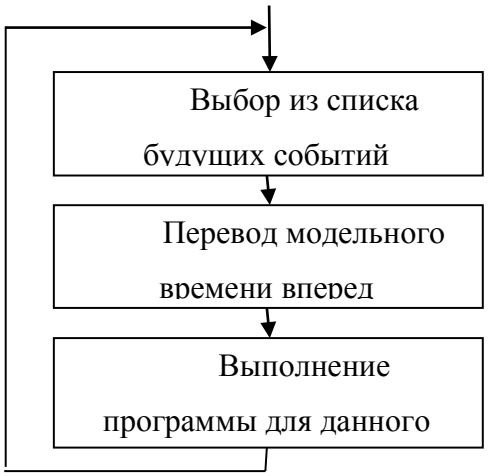

Рис. 4.2 Выполнение событий в ИМ.

4.2.3. Подход сканирования активностей

При использовании подхода сканирования активностей разработчик описывает все действия, в которых принимают участие элементы системы, и задает условия, определяющие начало и завершение действий (Рис. 4.3). После каждого продвижения имитационного времени условия всех возможных действий проверяются и если условие выполняется, то происходит имитация соответствующего действия. Выполнение действия приводит к изменению состояния системы и возможности выполнения новых действий. Например, для начала действия обработка детали на станке необходимо наличие свободной детали и наличие свободного станка. Если хотя бы одно из этих условий не выполнено, действие не начинается.

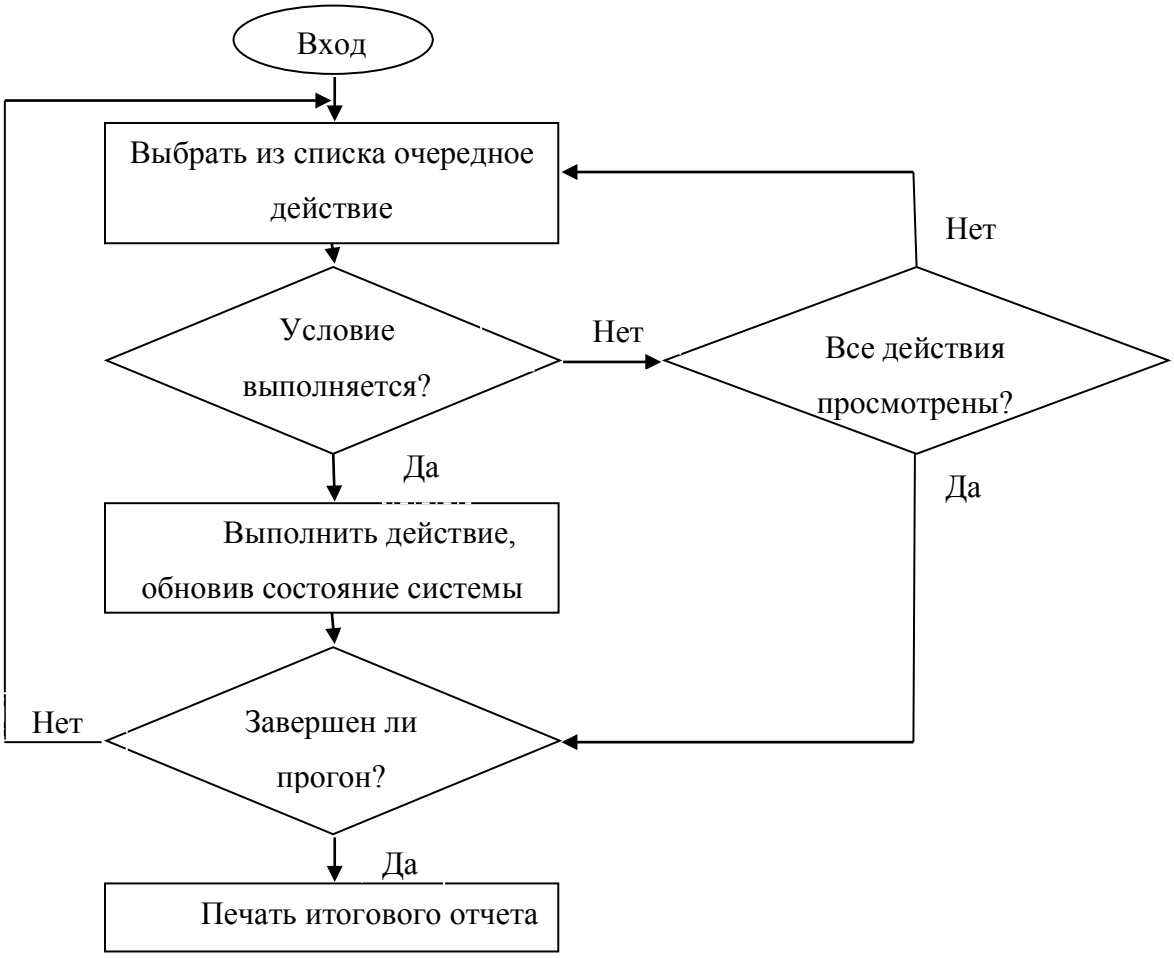

Рис. 4.3 Блок-схема реализации подхода сканирования.

4.3. Основные положения языка РДО

В основе системы РДО (1)– «Ресурсы, Действия, Операции» – лежат следующие положения:

Все элементы сложной дискретной системы (СДС) представлены как ресурсы, описываемые некоторыми параметрами.

Состояние ресурса определяется вектором значений всех его параметров; состояние СДС – значением всех параметров всех ресурсов.

Процесс, протекающий в СДС, описывается как последовательность целенаправленных действий и нерегулярных событий, изменяющих определенным образом состояния ресурсов; действия ограничены во времени двумя событиями: событиями начала и конца.

Нерегулярные события описывают изменение состояния СДС, непредсказуемые в рамках продукционной модели системы (влияние внешних по отношению к СДС факторов либо факторов, внутренних по отношению к ресурсам СДС). Моменты наступления нерегулярных событий случайны.

Действия описываются операциями, которые представляют собой модифицированные продукционные правила, учитывающие временные связи. Операция описывает предусловия, которым должно удовлетворять состояние участвующих в операции ресурсов, и правила изменения ресурсов в начале и конце соответствующего действия. При выполнении работ, связанных с созданием и использованием ИМ в среде РДО, пользователь оперирует следующими основными понятиями:

Модель - совокупность объектов РДО-языка, описывающих какой-то реальный объект, собираемые в процессе имитации показатели, кадры анимации и графические элементы, используемые при анимации, результаты трассировки.

Прогон - это единая неделимая точка имитационного эксперимента. Он характеризуется совокупностью объектов, представляющих собой исходные данные и результаты, полученные при запуске имитатора с этими исходными данными.

Проект - один или более прогонов, объединенных какой-либо общей целью. Например, это может быть совокупность прогонов, которые направлены на исследование одного конкретного объекта или выполнение одного контракта на имитационные исследования по одному или нескольким объектам.

Объект - совокупность информации, предназначенной для определенных целей и имеющая смысл для имитационной программы. Состав объектов обусловлен РДО-методом, определяющим парадигму представления СДС на языке РДО.

Объектами исходных данных являются:

- типы ресурсов (с расширением .rtp);

- ресурсы (с расширением .rss);
- образцы операций (с расширением .pat);
- операции (с расширением .opr);
- точки принятия решений (с расширением .dpt);
- константы, функции и последовательности (с расширением .fun);
- кадры анимации (с расширением .frm);
- требуемая статистика (с расширением .pmd);
- прогон (с расширением .smr).

Объекты, создаваемые РДО-имитатором при выполнении прогона:
- результаты (с расширением .pmv);
- трассировка (с расширением .trc).

4.4. Постановка задачи

Основная цель данного дипломного проекта – разработка языка процедурного программирования в систему имитационного моделирования РДО. Разрабатываемый язык программирования должен обеспечивать поддержку основных операторов, таких как:
- оператор присваивания;
- оператор объявления и инициализации локальных переменных;
- оператор выбора;
- циклический оператор;
- оператор выделения области локальной памяти;
- оператор возврата;
- оператор прерывания;
- оператор вызова подпрограммы;

Оператор объявления и инициализации локальных переменных, наиболее необходимый оператор для процедурного программирования. Возможность использования именованных переменных существенно упрощает процесс описания алгоритмов.

Оператор выбора позволяет описывать ветвления алгоритма по определённому логическому условию.

Циклический оператор удобен при необходимости описания повторения множества однотипных действий, которые должны повторно исполнятся в зависимости от какого-либо логического условия.

Оператор выделения области локальной памяти позволяет выделить пространство имен для локальных переменных, что подразумевает возможность использования одноименных переменных в разных областях локальной памяти.

Оператор возврата необходим для возвращения значения функции.

Оператор прерывания позволяет произвести выход из цикла до выполнения условия окончания.

Оператор вызова подпрограммы является оператором вызова функции.

5. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Система имитационного моделирования РДО безусловно является сложной и статически, и динамически. На это указывает сложная иерархическая структура системы со множеством различных связей между компонентами и ее сложное поведение во времени.

Ярко выраженная иерархическая структура и модульность системы определяют направление изучения системы сверху вниз. Т.е. мне необходимо применять принцип декомпозиции нужных модулей до тех пор, пока не будет достигнут уровень абстракции, представление на котором нужных объектов не нуждается в дальнейшей детализации для решения данной задачи.

5.1. Диаграмма компонентов

Для отображения зависимости между компонентами системы РДО и выделения среди них модернизируемых служит соответствующая диаграмма в нотации UML (3).

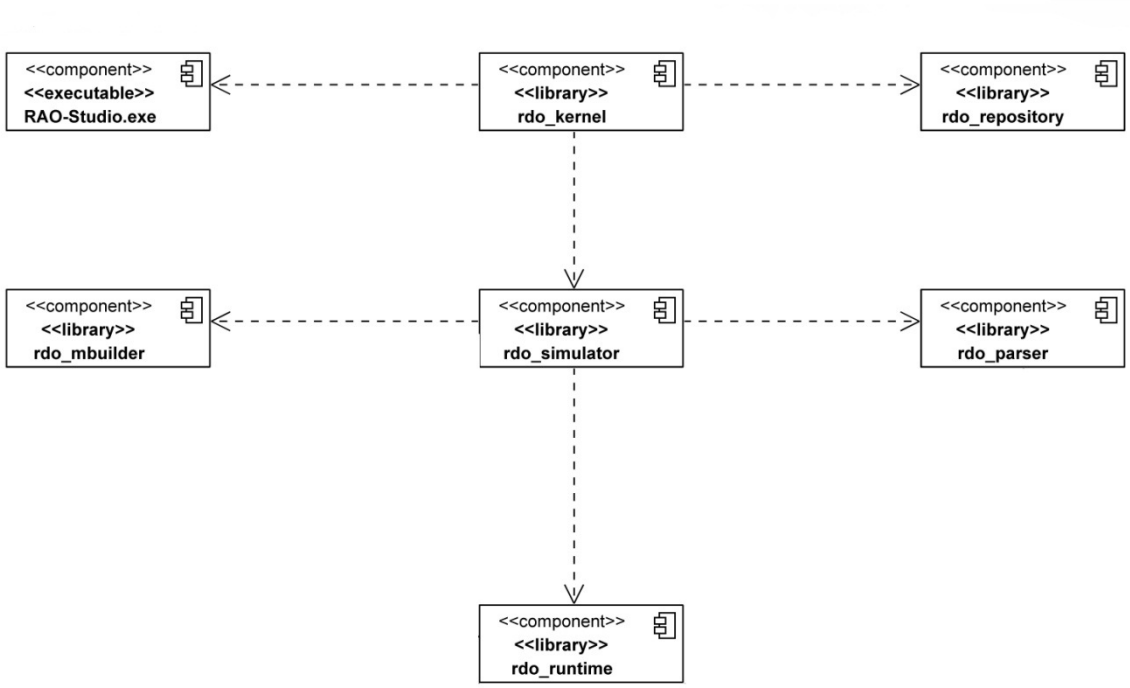

Рис. 5.1 Диаграмма компонентов.

Базовый функционал представленных на диаграмме компонентов (Рис. 5.1):

rdo_kernel реализует ядровые функции системы. Не изменяется при разработке системы.

RAO-studio.exe реализует графический интерфейс пользователя. Не изменяется при разработки системы.

rdo_repository реализует управление потоками данных внутри системы и отвечает за хранение и получение информации о модели. Не изменяется при разработке системы.

rdo_mbuilder реализует функционал, используемый для программного управления типами ресурсов и ресурсами модели. Не изменяется при разработке системы.

rdo_simulator управляет процессом моделирования на всех его этапах. Он осуществляет координацию и управление компонентами rdo_runtime и rdo_parser. Не изменяется при разработке системы.

rdo_parser производит лексический и синтаксический разбор исходных текстов модели, написанной на языке РДО. Модернизируется при разработке системы.

rdo_runtime отвечает за непосредственное выполнение модели, управление базой данных и базой знаний. Модернизируется при разработке системы.

Объекты компонента rdo_runtime инициализуруются при разборе исходного текста модели компонентом rdo_parser. Например, конструктор rdoParse:: RDORTPEnumParamType::constructorSuchAs содержит следующее выражение:

RDORTPEnumParamType* type = new RDORTPEnumParamType(parent(), enu, dv, such_as_src_info); которое выделяет место в свободной памяти и инициализирует объект rdoRuntime:: RDORTPEnumParamType, участвующий в дальнейшем процессе имитации.

В дальнейшем компоненты rdo_parser и rdo_runtime описываются более детально.

5.2. Структура логического вывода РДО

Логический вывод системы РДО представляет собой алгоритм, который определяет, какое событие в моделируемой системе должно произойти следующим в процессе имитации работы системы.

Во время имитации работы модели в системе существует одна МЕТА-логика. Она является контейнером для хранения разных логик. Сами логики являются контейнерами, в которых хранятся различные атомарные операции (например, нерегулярные события и правила). Таким образом, статическое представление БЗ модели на РДО представляет собой трехуровневое дерево, корнем которого является МЕТА-логика, а листьями - атомарные операции.

Интересно отметить, что реализация описанной структуры с помощью наследования – одного из основных механизмов объектно-ориентированного программирования – делает возможным на уровне логики работы РДО рекурсивное вложение логик внутрь логик. То есть архитектура имитатора РДО (rdo_runtime) не запрещает наличие точек принятия решений внутри точек принятия решений с любой глубиной вложенности.

Поиск активности, которая должна быть запущена следующей, начинается с обращения класса RDOSimulator к своему атрибуту m_logics, в котором хранится описанная выше МЕТА-логика. Далее от корня дерева к листьям распространяется волна вызовов метода onCheckCondition(). Т.е. onCheckCondition() вызывается у МЕТА-логики, затем циклически у ее логик, и наконец, циклически проверяются все атомарные операции каждой логики. Как только найдена активность, которая может быть выполнена, происходит ее кэширование (запоминание) внутри логики и кэширование самой логики внутри МЕТА-логики. После этого управление снова передается в RDOSimulator и найденная активность выполняется.

Для управления поиском очередной активности с помощью приоритетов точек принятия решений необходимо отсортировать список логик внутри МЕТА-логики по убыванию приоритета и в дальнейшем производить поиск в отсортированном списке.

6. РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ НА СИСТЕМУ

6.1. Основания для разработки

- Задание на дипломный проект.

6.2. Назначение разработки

Основная цель данного дипломного проекта – реализовать поддержку языка процедурного программирования в системе имитационного моделирования РДО, а так же разработать классы и методы, реализующие поддержку языка процедурного программирования во внутренней структуре RAO Studio.

6.3. Требования к программе или программному изделию

6.3.1. Требования к функциональным характеристикам

Требования к составу выполняемых функций реализуемой системы заключаются в возможности использования модулей на процедурном языке программирования при моделировании в системе РДО. Система моделирования РДО должна выполнять следующие функции:

- Поддержка синтаксиса описания модулей на процедурном языке программирования при описании объектов языка РДО.
- Создание и использование «локальных переменных».
- Поддержка синтаксиса операторов «{» и «}», используемых для выделения области локальной памяти.
- Поддержка синтаксиса оператора «for», используемого для описания циклов.
- Поддержка синтаксиса оператора «break», использующегося для выхода из циклов.
- Поддержка синтаксиса оператора «return», используемого для возвращения значения при описании функций.
- Создание типа данных «массив» не ограниченной размерности.
- Создание ресурсов типа «массив» не ограниченной размерности.
- Проверка синтаксической верности описания массивов.
- Ввод массивов при анимации модели в текстовом виде.

6.3.2. Требования к надежности

Основное требование к надежности направлено на поддержание в исправном и работоспособном ЭВМ на которой происходит использование программного комплекса RAO-Studio.

6.3.3. Условия эксплуатации

- Эксплуатация должна производиться на оборудовании, отвечающем требованиями к составу и параметрам технических средств, и с применением программных средств, отвечающим требованиям к программной совместимости.
- Аппаратные средства должны эксплуатироваться в помещениях с выделенной розеточной электросетью 220В ±10%, 50 Гц с защитным заземлением.

6.3.4. Требования к составу и параметрам технических средств

Программный продукт должен работать на компьютерах со следующими характеристиками:

- объем ОЗУ не менее 256 Мб;
- объем жесткого диска не менее 20 Гб;
- микропроцессор с тактовой частотой не менее 400 МГц;
- монитор не менее 15” с разрешением от 800*600 и выше;

6.3.5. Требования к информационной и программной совместимости

Данная система должна работать под управлением операционных систем Windows 2000, Windows XP, Windows Vista и Windows 7.

6.3.6. Требования к маркировке и упаковке

Не предъявляются.

6.3.7. Требования к транспортированию и хранению

Не предъявляются.

6.4. Требования к программной документации

Не предъявляются.

6.5. Стадии и этапы разработки

- Предпроектное исследование.
- Концептуальный этап проектирования.
- Технический этап проектирования.
- Рабочий этап проектирования.

6.6. Порядок контроля и приемки

Контроль и приемка поддержки языка процедурного программирования должны осуществляться в процессе проверки функциональности (апробирования) системы имитационного моделирования на тестовом примере модели в соответствии с требованиями к функциональным характеристикам системы.

7. ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

7.1. Разработка синтаксиса описания типа ресурса массив

Типы ресурсов определяют структуру глобальной базы данных программы (модели) и их описывают в отдельном объекте (имеет расширение **.rtp**).

Описание каждого типа ресурса имеет следующий формат:

```
$Resourse_type<имя_типа>:<вид_ресурсов>
$Parameters
<описание_параметра>{< описание_параметра >}
$End
```

имя_типа

Имя типа представляет собой простое имя. Имена типов должны быть различными для всех типов и не должны совпадать с предопределенными и ранее использованными именами.

вид_ресурсов

Вид ресурсов данного типа может быть одним из следующих:

permanent: Постоянные ресурсы; ресурсы этого вида всегда присутствуют в модели, они не могут быть уничтожены или созданы во время прогона

temporary: Временные ресурсы; ресурсы этого вида могут во время прогона создаваться и уничтожаться при выполнении операций, правил и совершении нерегулярных событий

описание_параметра

Описание параметра ресурса имеет формат:

```
< имя_параметра>:< тип_параметра >[=< значение_по_умолчанию>]
```

имя_параметра

Имя параметра - это **простое имя**. Имена параметров должны быть различными для всех параметров данного типа и не должны совпадать предопределенными и ранее использованными именами. Имя параметра может совпадать с именем параметра другого типа ресурсов.

тип_параметра

Тип параметра - это один из возможных *типов данных языка*. Ссылки возможны на параметры ранее описанных типов ресурсов и на ранее описанные параметры данного типа ресурсов.

значение_по_умолчанию

Для параметра любого типа может быть задано значение по умолчанию. Это значение указывают после знака равенства целой или вещественной численной константой, либо именем значения для перечислимого параметра. При указании типа ссылкой также возможно задание значения по умолчанию. При этом задаваемое значение может отличаться от значения по умолчанию того параметра, на тип которого проводится ссылка.

Тип данных языка РДО

- целый тип - **integer**;
- вещественный тип - **real**;
- строковый тип – **string;**
- логический тип – **bool** ;
- перечислимый тип;
- массив – **array<** *Тип данных языка РДО***>;**
- ссылка на один из выше определенных типов - **such_as**.

7.2. Разработка синтаксиса описания значения массива

Значения параметров ресурса задают в позиционном соответствии с порядком следования параметров в описании типа. Значения задают целой или вещественной численной константой либо именем значения в соответствии с типом параметра.

Численные константы записывают так же, как в языках программирования высокого уровня.

Константы целого типа представляют собой последовательность цифр, перед которой может стоять знак "+" или "-".

Вещественные константы состоят из целой и дробной частей, разделенных точкой, за которыми может следовать порядок. Порядок числа начинается символом "E" или "e", за которым следует значение порядка (целая константа со знаком). Вещественная константа также может иметь знак.

Значение массива задается в формате: «[», Константа одного из типов данных языка РДО, «]». Поскольку константа одного из типов данных языка РДО может являться массивом, это позволяет описывать значения многомерных массивов.

7.3. Разработка синтаксиса объявления и инициализации локальной переменной

При объявлении локальной переменной сначала указывается тип данных локальной переменной, который должен являться одним из зарегистрированных типов данных языка РДО. Затем указывается имя данной переменной и, опционно, оператор присваивания со значением по умолчанию.

7.4. Разработка синтаксиса описания циклического оператора

При описании циклического оператора указывается ключевое слово «for», после него в скобках последовательно указывается: оператор присваивания начального значения циклической переменной или объявления локальной циклической переменной с инициализирующим значением, логическое условие выполнения текущей итерации цикла и выражение приращения циклической переменной, разделённые точкой с запятой. После закрывающей скобки указывается тело цикла, состоящее из одной операции или списка операций заключенных в фигурные скобки.

7.5. Разработка архитектуры компонента rdo_parser

Для возможности обработки новой конструкции в коде модели требуют изменений лексический и синтаксический анализаторы РДО, также требуется добавить классы (4), реализующие поддержку описания массивов и разрабатываемых операторов процедурного программирования.

7.6. Разработка архитектуры компонента rdo_runtime

В пространстве имен rdoRuntime необходимо добавить классы описывающие массивы, а так же классы и методы, реализующие логику работы операторов процедурного программирования.

8. РАБОЧЕЕ ПРОЕКТИРОВАНИЕ

8.1. Изменения в файлах синтаксического анализатора

Для реализации в среде имитационного моделирования нового инструмента разработанного на концептуальном и техническом этапах проектирования, в первую очередь необходимо добавить новые термальные символы в лексический анализатор РДО и нетермальные символы в грамматический анализатор.

8.1.1. Синтаксический анализ объявления типа данных массив

В лексическом анализаторе (flex) я добавил новый токен *RDO_array,* который может быть записан единственным способом:

```
array        return(RDO_array);
```

Этот токен необходимо также добавить в генератор синтаксического анализатора (bison):

```
%token       RDO_array
```

Далее необходимо добавить описание типа массива и описание значения по умолчанию:

```
param_type_array param_value_default

param_type_array
  :RDO_array '<' param_type '>'

param_value_default:     /* empty */
  | '=' param_value
  | '=' error

param_value
  :RDO_INT_CONST
  | RDO_REAL_CONST
  | RDO_STRING_CONST
  | RDO_IDENTIF
  | RDO_BOOL_CONST
  | param_array_value

param_array_value
  :'[' array_item ']'
  |'[' array_item error

array_item
  :param_value
  |array_item ',' param_value
  |array_item param_value
```

Из этого кода можно сделать вывод, что типом данных содержащихся в массиве могут быть массивы. Это даёт возможность создавать многомерные массивы неограниченной вложенности.

8.1.2. Синтаксический анализ объявления операторов процедурного программирования

В файлы синтаксического анализатора было добавлено описание операторов процедурного программирования.

```
statement
  : empty_statement ';'
  | empty_statement error
  | equal_statement ';'
  | equal_statement error
  | set_array_item_statement ';'
  | set_array_item_statement error
  | local_variable_declaration ';'
  | local_variable_declaration error
  | if_statement
  | for_statement
  | return_statement ';'
  | return_statement error
  | break_statement ';'
  | break_statement error
  | open_brace statement_list close_brace
```

8.1.2.1. Синтаксический анализ объявления оператора присваивания

В файлы синтаксического анализатора было добавлено описание оператора присваивания.

```
equal_statement
  : RDO_IDENTIF increment_or_decrement_type
  | RDO_IDENTIF param_equal_type fun_arithm
  | RDO_IDENTIF param_equal_type error
  | RDO_IDENTIF error fun_arithm

increment_or_decrement_type
  : RDO_IncrEqual
  | RDO_DecrEqual

param_equal_type
  : RDO_set
  | '='
  | RDO_PlusEqual
  | RDO_MinusEqual
  | RDO_MultiplyEqual
  | RDO_DivideEqual
```

8.1.2.2. Синтаксический анализ оператора записи значения массива

В файлы синтаксического анализатора было добавлено описание оператора записи значания элемента массива.

```
set_array_item_statement
  : RDO_IDENTIF '[' fun_arithm ']' '=' fun_arithm
```

8.1.2.3. Синтаксический анализ объявления и инициализации локальной переменной

В файлы синтаксического анализатора было добавлено описание объявления и инициализации локальной переменной.

```
local_variable_declaration
 : type_declaration init_declaration_list

type_declaration
 : RDO_integer
 | RDO_real
 | RDO_string
 | param_type_array
 | RDO_bool
 | param_type_enum
 | param_type_such_as

init_declaration_list
 : init_declaration
 | init_declaration_list ',' init_declaration

init_declaration
 : RDO_IDENTIF
 | RDO_IDENTIF '=' fun_arithm
```

8.1.2.4. Синтаксический анализ объявления циклического оператора

В лексическом анализаторе (flex) я добавил новый токен *RDO_array,* который может быть записан единственным способом:

```
for          return(RDO_for);
```

Этот токен необходимо также добавить в генератор синтаксического анализатора (bison):

```
%token       RDO_for
```

В файлы синтаксического анализатора было добавлено описание циклического оператора.

```
for_statement
 : RDO_for '(' local_variable_declaration ';' fun_logic ';' equal_statement
')' statement
 | RDO_for '(' equal_statement ';' fun_logic ';' equal_statement ')'
statement
```

8.1.2.5. Синтаксический анализ объявления оператора возврата

В лексическом анализаторе (flex) я добавил новый токен *RDO_array,* который может быть записан единственным способом:

```
return       return(RDO_return);
```

Этот токен необходимо также добавить в генератор синтаксического анализатора (bison):

```
%token       RDO_return
```

В файлы синтаксического анализатора было добавлено описание циклического оператора.

```
return_statement
 :RDO_Return fun_arithm
```

8.1.2.6. Синтаксический анализ объявления оператора прерывания

В лексическом анализаторе (flex) я добавил новый токен *RDO_array,* который может быть записан единственным способом:

```
break        return(RDO_break);
```

Этот токен необходимо также добавить в генератор синтаксического анализатора (bison):

```
%token       RDO_break
```

В файлы синтаксического анализатора было добавлено описание циклического оператора.

```
break_statement
 :RDO_Break
```

8.2. Изменения в пространстве имен rdoRuntime

8.2.1. Объявление класса RDOArrayType

Класс предназначен для описания типа массива и содержит в себе информацию о типах элементов содержащихся в массивах.

```
class RDOArrayType: public RDOType
{
DECLARE_FACTORY(RDOArrayType);
public:
typedef LPRDOType LPItemType;

LPItemType getItemType() const;

private:
RDOArrayType(CREF(LPItemType) pItemType);

LPItemType m_pItemType;
};
```

Объявление класса RDOArrayValue

Класс предназначен для описания массива и содержит в себе информацию о типе массива и элементы массива.

```
class RDOArrayValue
{
public:
 typedef std::vector<RDOValue> Container;

 RDOArrayValue(CREF(LPRDOArrayType) pType);
 RDOArrayValue(CREF(RDOArrayValue)  value);
 ~RDOArrayValue();

 CREF(LPRDOArrayType) type() const;
```

```
 void insertItem(CREF(RDOValue) pArray);
 Container::iterator containerBegin();
 Container::iterator containerEnd();
 void insertItems(Container::iterator itr, Container::iterator itrFst,
Container::iterator itrLst);
 void  eraseItems(Container::iterator itrFst, Container::iterator itrLst
);
 CREF(RDOValue) operator[] (CREF(RDOValue) ind) const;

 ruint arraySize() const;

 tstring getAsString() const;
 void setArrayItem(CREF(RDOValue) ind, CREF(RDOValue) item);

private:
 Container       m_container;
 LPRDOArrayType  m_pArrayType;
};
```

8.2.2. Объявление класса RDOCalcArraySize, RDOCalcArrayItem, RDOCalcSetArrayItem

Данные классы предназначены для описания операторов возвращения размера массива, возвращения значения элемента массива по индексу и записи значения элемента массива с указанным индексом соответственно.

```
CALC(RDOCalcArraySize)
{
DECLARE_FACTORY(RDOCalcArraySize)
private:
 RDOCalcArraySize(CREF(LPRDOCalc) pArray);

 LPRDOCalc m_pArray;

 DECALRE_ICalc;
};

CALC(RDOCalcArrayItem)
{
DECLARE_FACTORY(RDOCalcArrayItem)
private:
 RDOCalcArrayItem(CREF(LPRDOCalc) pArray, CREF(LPRDOCalc) pArrayInd);

 LPRDOCalc m_pArray;
 LPRDOCalc m_pArrayInd;

 DECALRE_ICalc;
};

CALC(RDOCalcSetArrayItem)
{
 DECLARE_FACTORY(RDOCalcSetArrayItem)
private:
 RDOCalcSetArrayItem(CREF(LPRDOCalc) pArray, CREF(LPRDOCalc) pArrayInd,
CREF(LPRDOCalc) pSetItem);

 LPRDOCalc m_pArray;
 LPRDOCalc m_pArrayInd;
 LPRDOCalc m_pSetItem;

 DECALRE_ICalc;
};
```

Основные методы класса, doCalc, ализующи логику работы данных операторов.

- Данный метод возвращает размер массива.

```
REF(RDOValue) RDOCalcArraySize::doCalc(PTR(RDORuntime) pRuntime)
{
 CREF(RDOArrayValue) arrayValue = m_pArray->calcValue(pRuntime).getArray();
 m_value = RDOValue(arrayValue.arraySize());
 return m_value;
}
```

- Данный метод возвращает значение элемента массива по индексу.

```
REF(RDOValue) RDOCalcArrayItem::doCalc(PTR(RDORuntime) pRuntime)
{
 CREF(RDOArrayValue) arrayValue = m_pArray->calcValue(pRuntime).getArray();
 m_value = arrayValue[m_pArrayInd->calcValue(pRuntime)];
 return m_value;
}
```

- Данный метод записывает значение элемента массива по индексу и возвращает изменнёный массив.

```
REF(RDOValue) RDOCalcSetArrayItem::doCalc(PTR(RDORuntime) pRuntime)
{
 m_value = m_pArray->calcValue(pRuntime);
 m_value.setArrayItem(m_pArrayInd->calcValue(pRuntime), m_pSetItem-
>calcValue(pRuntime));
 return m_value;
}
```

8.2.3. Объявление класса RDOCalcFor

Класс пердназначен для описания циклического оператора.

```
CALC(RDOCalcFor)
{
 DECLARE_FACTORY(RDOCalcFor)
private:
 RDOCalcFor(CREF(LPRDOCalc) pDeclaration, CREF(LPRDOCalc) pCondition,
CREF(LPRDOCalc) pExpression, CREF(LPRDOCalc) pStatement);

 LPRDOCalc m_pDeclaration;
 LPRDOCalc m_pCondition;
 LPRDOCalc m_pExpression;
 LPRDOCalc m_pStatement;

 DECALRE_ICalc;
};
```

Основной метод класса doCalc, ализующий логику работы данного оператора. В данном методе реализовано повторения выполнения операций содержащихся в теле цикла и проверка значения флагов прерывания и возврата.

```
REF(RDOValue) RDOCalcFor::doCalc(PTR(RDORuntime) pRuntime)
{
 if(pRuntime->getFunBreakFlag() == RDORuntime::FBF_CONTINUE)
 {
       m_value = m_pDeclaration->calcValue(pRuntime);
       while (m_pCondition->calcValue(pRuntime).getAsBool())
       {
```

```
                m_value = m_pStatement->calcValue(pRuntime);
                m_pExpression->calcValue(pRuntime);
                if(pRuntime->getFunBreakFlag() == RDORuntime::FBF_BREAK)
                {
                      pRuntime->setFunBreakFlag(RDORuntime::FBF_CONTINUE);
                      return m_value;
                }
                if(pRuntime->getFunBreakFlag() == RDORuntime::FBF_RETURN)
                {
                      return m_value;
                }
          }
     }
     return m_value;
}
```

8.2.4. Объявление классов RDOCalcIf и RDOCalcIfElse

Классы предназначены для описания оператора выбора без ветвления инче и светвлением соответственно.

```
CALC(RDOCalcIf)
{
 DECLARE_FACTORY(RDOCalcIf)
private:
 RDOCalcIf(CREF(LPRDOCalc) pCondition, CREF(LPRDOCalc) pStatement);

 LPRDOCalc m_pCondition;
 LPRDOCalc m_pStatement;

 DECALRE_ICalc;
};

CALC(RDOCalcIfElse)
{
 DECLARE_FACTORY(RDOCalcIfElse)
private:
 RDOCalcIfElse(CREF(LPRDOCalc) pCondition, CREF(LPRDOCalc) pIfStatement,
CREF(LPRDOCalc) pElseStatement);

 LPRDOCalc m_pCondition;
 LPRDOCalc m_pIfStatement;
 LPRDOCalc m_pElseStatement;

 DECALRE_ICalc;
};
```

Основные методы класса, doCalc, ализующии логику работы данных операторов.

- Данный метод реализует выполнение оператора под условием, оператор выполняется только в том случае если условие истинно.

```
REF(RDOValue) RDOCalcIf::doCalc(PTR(RDORuntime) pRuntime)
{
 m_value = RDOValue(false);
 return (m_pCondition->calcValue(pRuntime).getAsBool()) ? m_pStatement-
>calcValue(pRuntime) : (m_value);
}
```

- Данный метод реализует выполнение оператора под условием или оператор в блоке «иначе», в зависимости от условия.

```
REF(RDOValue) RDOCalcIfElse::doCalc(PTR(RDORuntime) pRuntime)
{
 return (m_pCondition->calcValue(pRuntime).getAsBool()) ? m_pIfStatement-
>calcValue(pRuntime) : m_pElseStatement->calcValue(pRuntime);
}
```

8.2.5. Объявление класса RDOCalcFunReturn

Данный класс предназначен для описания оператора возврата.

```
CALC(RDOCalcFunReturn)
{
 DECLARE_FACTORY(RDOCalcFunReturn)
private:
 RDOCalcFunReturn(CREF(LPRDOCalc) pReturn);

 LPRDOCalc m_pReturn;
 DECALRE_ICalc;
};
```

Основной метод класса, doCalc, ралізующий логику работы данного оператора. Данный метод возвращает значение выражения указанного в операторе возврата и устанавливает значение флага в `FBF_RETURN`.

```
REF(RDOValue) RDOCalcFunReturn::doCalc(PTR(RDORuntime) pRuntime)
{
 m_value = m_pReturn->calcValue(pRuntime);
 pRuntime->setFunBreakFlag(RDORuntime::FBF_RETURN);
 return m_value;
}
```

8.2.6. Объявление класса RDOCalcFunBreak

Данный класс предназначен для описания оператора преравания.

```
CALC(RDOCalcFunBreak)
{
 DECLARE_FACTORY(RDOCalcFunBreak)
private:
 RDOCalcFunBreak();

 DECALRE_ICalc;
};
```

Основной метод класса doCalc, ралізующий логику работы данного оператора. Данный метод устанавливает значение флага в `FBF_BREAK`.

```
REF(RDOValue) RDOCalcFunBreak::doCalc(PTR(RDORuntime) pRuntime)
{
 pRuntime->setFunBreakFlag(RDORuntime::FBF_BREAK);
 return m_value;
}
```

8.2.7. Объявление классов RDOCalcCreateLocalVariable, RDOCalcInitLocalVariable, RDOCalcGetLocalVariable, RDOCalcSetLocalVariable

Данные класы предназначены для создания локальной переменной в памяти, инициализации её начальным значением, получения значения лакальной переменной и записи значения локальной переменной соответсвенно.

```
CALC(RDOCalcCreateLocalVariable)
{
DECLARE_FACTORY(RDOCalcCreateLocalVariable)
private:
 RDOCalcCreateLocalVariable(CREF(tstring) name);

 tstring m_name;

 DECALRE_ICalc;
};
```

```
CALC(RDOCalcInitLocalVariable)
{
DECLARE_FACTORY(RDOCalcInitLocalVariable)
private:
 RDOCalcInitLocalVariable(CREF(tstring) name, CREF(LPRDOCalc) pCalc);

 tstring   m_name;
 LPRDOCalc m_pCalc;

 DECALRE_ICalc;
};

CALC(RDOCalcGetLocalVariable)
{
DECLARE_FACTORY(RDOCalcGetLocalVariable)
private:
 RDOCalcGetLocalVariable(CREF(tstring) name);

 tstring m_name;

 DECALRE_ICalc;
};
```

Класс для записи значения локальной переменной является шаблонным классом. Это необхадимо, поскольку от типа оператора присваивания зависит логика работы основного метода класса.

```
template <EqualType equalType>
class RDOCalcSetLocalVariable: public RDOCalc
{
DECLARE_FACTORY(RDOCalcSetLocalVariable)
private:
 RDOCalcSetLocalVariable(CREF(tstring) name, LPRDOCalc pCalc = NULL);
 virtual ~RDOCalcSetLocalVariable();
```

```
 tstring   m_name;
 LPRDOCalc m_pCalc;

 DECALRE_ICalc;
};
```

Основные методы классов, doCalc, ализующии логику работы данных операторов.

- Данный метод создаёт в локальной памяти не проинициализированную локальную переменную.

```
REF(RDOValue) RDOCalcCreateLocalVariable::doCalc(PTR(RDORuntime) pRuntime)
{
 pRuntime->getMemoryStack()->create(m_name);
 m_value = RDOValue();
 return m_value;
}
```

- Данный метод инициализирует локальную переменную.

```
REF(RDOValue) RDOCalcInitLocalVariable::doCalc(PTR(RDORuntime) pRuntime)
{
 pRuntime->getMemoryStack()->set(m_name, m_pCalc->calcValue(pRuntime));
 return m_value;
}
```

- Данный метод возвращает значение локальной переменной по имени.

```
REF(RDOValue) RDOCalcGetLocalVariable::doCalc(PTR(RDORuntime) pRuntime)
{
 m_value = pRuntime->getMemoryStack()->get(m_name);
 return m_value;
}
```

- Данный метод сохраняет в памяти значение локальной переменной.

```
template <>
inline REF(RDOValue)
RDOCalcSetLocalVariable<ET_EQUAL>::doCalc(PTR(RDORuntime) pRuntime)
{
 pRuntime->getMemoryStack()->set(m_name, m_pCalc->calcValue(pRuntime));
 return m_value;
}
```

- Данный метод увеличивает значение на указанную велечину и сохраняет в памяти значение локальной переменной.

```
template <>
inline REF(RDOValue)
RDOCalcSetLocalVariable<ET_PLUS>::doCalc(PTR(RDORuntime) pRuntime)
{
 rdoRuntime::RDOValue pValue = pRuntime->getMemoryStack()->get(m_name) +
m_pCalc->calcValue(pRuntime);
```

```
 pRuntime->getMemoryStack()->set(m_name, pValue);
 return m_value;
}
```

- Данный метод уменьшает значение на указанную велечину и сохраняет в памяти значение локальной переменной.

```
template <>
inline REF(RDOValue)
RDOCalcSetLocalVariable<ET_MINUS>::doCalc(PTR(RDORuntime) pRuntime)
{
 rdoRuntime::RDOValue pValue = pRuntime->getMemoryStack()->get(m_name) -
m_pCalc->calcValue(pRuntime);
 pRuntime->getMemoryStack()->set(m_name, pValue);
 return m_value;
}
```

- Данный метод увеличивает значение в указанное число раз и сохраняет в памяти значение локальной переменной.

```
template <>
inline REF(RDOValue)
RDOCalcSetLocalVariable<ET_MULTIPLY>::doCalc(PTR(RDORuntime) pRuntime)
{
 rdoRuntime::RDOValue pValue = pRuntime->getMemoryStack()->get(m_name) *
m_pCalc->calcValue(pRuntime);
 pRuntime->getMemoryStack()->set(m_name, pValue);
 return m_value;
}
```

- Данный метод уменьшает значение в указанное число раз и сохраняет в памяти значение локальной переменной.

```
template <>
inline REF(RDOValue)
RDOCalcSetLocalVariable<ET_DIVIDE>::doCalc(PTR(RDORuntime) pRuntime)
{
 rdoRuntime::RDOValue pValue = pRuntime->getMemoryStack()->get(m_name) /
m_pCalc->calcValue(pRuntime);
 pRuntime->getMemoryStack()->set(m_name, pValue);
 return m_value;
}
```

- Данный метод увеличивает значение на единицу и сохраняет в памяти значение локальной переменной.

```
template <>
inline REF(RDOValue)
RDOCalcSetLocalVariable<ET_INCR>::doCalc(PTR(RDORuntime) pRuntime)
{
 rdoRuntime::RDOValue pValue = pRuntime->getMemoryStack()->get(m_name) +
RDOValue(1);
 pRuntime->getMemoryStack()->set(m_name, pValue);
 return m_value;
}
```

- Данный метод уменьшает значение на единицу и сохраняет в памяти значение локальной переменной.

```
template <>
inline REF(RDOValue)
RDOCalcSetLocalVariable<ET_DECR>::doCalc(PTR(RDORuntime) pRuntime)
{
 rdoRuntime::RDOValue pValue = pRuntime->getMemoryStack()->get(m_name) -
RDOValue(1);
 pRuntime->getMemoryStack()->set(m_name, pValue);
 return m_value;
}
```

8.2.8. Объявление перечесления EqualType

Данное перечесление необходимо для шаблонного класса записи значения локальной переменной. В данном перечислении описаны возможные типы оператора присваивания.

```
enum EqualType
{
 ET_UNDEFINED,
 ET_NOCHANGE,
 ET_EQUAL,
 ET_PLUS,
 ET_MINUS,
 ET_MULTIPLY,
 ET_DIVIDE,
 ET_INCR,
 ET_DECR
};
```

8.2.9. Объявление класса RDOCalcOpenBrace, RDOCalcCloseBrace, RDOCalcFunEnd

Данные классы предназначены для открытия и закрытия области локальной памяти.

```
CALC(RDOCalcOpenBrace)
{
 DECLARE_FACTORY(RDOCalcOpenBrace)
private:
 RDOCalcOpenBrace();

 DECALRE_ICalc;
};

CALC(RDOCalcCloseBrace)
{
 DECLARE_FACTORY(RDOCalcCloseBrace)
private:
 RDOCalcCloseBrace();

 DECALRE_ICalc;
};
```

Данный класс разработан для закрытия области локальной памяти тела функции.

```
CALC(RDOCalcFunEnd)
{
 DECLARE_FACTORY(RDOCalcFunEnd)
```

```
private:
 RDOCalcFunEnd();

 DECALRE_ICalc;
};
```

Основные методы классов, doCalc, ализующии логику работы данных операторов.

- Данный метод создает экземпляр локальной памяти и помещает его в стек памяти.

```
REF(RDOValue) RDOCalcOpenBrace::doCalc(PTR(RDORuntime) pRuntime)
{
 LPRDOMemory pLocalMemory = rdo::Factory<RDOMemory>::create();
 pRuntime->getMemoryStack()->push(pLocalMemory);
 return m_value;
}
```

- Данный метод удаляет экземпляр локальной памяти из стека памяти.

```
REF(RDOValue) RDOCalcCloseBrace::doCalc(PTR(RDORuntime) pRuntime)
{
 pRuntime->getMemoryStack()->pop();
 return m_value;
}
```

- Данный метод удаляет экземпляр локальной памяти из стека памяти и переключает значение флага прерывания и возврата в FBF_CONTINUE.

```
REF(RDOValue) RDOCalcFunEnd::doCalc(PTR(RDORuntime) pRuntime)
{
 pRuntime->getMemoryStack()->pop();
 pRuntime->setFunBreakFlag(RDORuntime::FBF_CONTINUE);
 return m_value;
}
```

8.2.10. Объявление класса RDOCalcBodyBrace, RDOCalcFunBodyBrace

Данные класы предназначены для описания списков операций при описании образцов операций и при описании алгоритмической функции соответственно.

```
CALC(RDOCalcBodyBrace)
{
 DECLARE_FACTORY(RDOCalcBodyBrace)
public:
 typedef std::vector<LPRDOCalc> CalcList;

 void addCalc(CREF(LPRDOCalc) pCalc);

private:
 RDOCalcBodyBrace();

 CalcList m_calcList;

 DECALRE_ICalc;
};

CALC_SUB(RDOCalcFunBodyBrace, RDOFunCalc)
{
 DECLARE_FACTORY(RDOCalcFunBodyBrace)
```

```
public:
 typedef std::vector<LPRDOCalc> CalcFunList;

 void addFunCalc(CREF(LPRDOCalc) pCalc);
 void addRetCalc(CREF(LPRDOCalc) pCalc);

private:
 RDOCalcFunBodyBrace();

 CalcFunList m_calcFunList;

 DECALRE_ICalc;
};
```

Основные методы классов, doCalc, ализующии логику работы данных операторов.

- Данй метод реализует последовательное ваполнение всех оперций из списка операций.

```
REF(RDOValue) RDOCalcBodyBrace::doCalc(PTR(RDORuntime) pRuntime)
{
 STL_FOR_ALL(m_calcList, calc_it)
 {
       (*calc_it)->calcValue(pRuntime);
 }

 m_value = RDOValue(m_calcList.size());
 return m_value;
}
```

- Данй метод реализует последовательное ваполнение всех оперций из списка операций с учётом обработки значения флага прерывания и возврата.

```
REF(RDOValue) RDOCalcFunBodyBrace::doCalc(PTR(RDORuntime) pRuntime)
{
 if (pRuntime->getFunBreakFlag() == RDORuntime::FBF_CONTINUE)
 {
       STL_FOR_ALL(m_calcFunList, calcIt)
       {
             LPRDOCalc pCalc = *calcIt;
             ASSERT(pCalc);
             m_value = pCalc->calcValue(pRuntime);
             if (pRuntime->getFunBreakFlag() == RDORuntime::FBF_BREAK)
             {
                   m_calcFunList.back()->calcValue(pRuntime);
                   pRuntime->setFunBreakFlag(RDORuntime::FBF_CONTINUE);
                   m_value = RDOValue(false);
                   return m_value;
             }
             if (pRuntime->getFunBreakFlag() == RDORuntime::FBF_RETURN)
             {
                   m_calcFunList.back()->calcValue(pRuntime);
                   return m_value;
             }
       }
 }
 return m_value;
}
```

8.3. Изменения в пространстве имен rdoParser

8.3.1. Объявление класса RDOArrayType

Класс предназначен для описания типа массива и содержит в себе информацию о положении описания типа массива в тексте модели и типе элементов содержащихся в массиве.

```
class RDOArrayType: public RDOType, public RDOParserSrcInfo
{
DECLARE_FACTORY(RDOArrayType);
public:
 CREF(LPTypeInfo)           getItemType        () const;
 rdoRuntime::LPRDOArrayType getRuntimeArrayType() const;
 LPTypeInfo                 typeInfo           () const;

private:
 RDOArrayType         (CREF(LPTypeInfo) pItemType, CREF(RDOParserSrcInfo)
src_info);
 virtual ~RDOArrayType();

 LPTypeInfo m_pItemType;

 DECLARE_IType;
 DECLARE_IModelStructure;
};
DECLARE_POINTER(RDOArrayType)
```

8.3.2. Объявление класса RDOArrayValue

Класс предназначен для описания массива и содержит информацию о типе массива и элементы массива.

```
OBJECT(RDOArrayValue) IS INSTANCE_OF(RDOParserSrcInfo)
{
DECLARE_FACTORY(RDOArrayValue);
public:
 typedef std::vector<RDOValue> Container;

 void insertItem(CREF(RDOValue) value);

 CREF(LPRDOArrayType) getArrayType() const;
  REF(LPRDOArrayType) getArrayType();
 rdoRuntime::RDOValue getRArray   () const;
 tstring              getAsString () const;
 CREF(Container)      getContainer() const;

private:
 RDOArrayValue(CREF(LPRDOArrayType) pArrayType);
 virtual ~RDOArrayValue();

 Container      m_container;
 LPRDOArrayType m_pArrayType;
};
```

8.3.3. Объявление класса VariableContainer

Класс предназначен для хранения списка локальных переменных.

```
OBJECT(VariableContainer)
{
DECLARE_FACTORY(VariableContainer);
public:
 CREF(rdoRuntime::LPRDOCalc) getCalc        () const;
 CREF(LPLocalVariable)       getLocalVariable() const;

private:
 VariableContainer(CREF(rdoRuntime::LPRDOCalc) pCalc, CREF(LPLocalVariable)
pLocalVariable);

 rdoRuntime::LPRDOCalc m_pCalc;
 LPLocalVariable       m_pLocalVariable;
};
```

9. ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

В рамках данного дипломного проекта разрабатывался язык процедурного программирования в системе имитационного моделирования РДО. В процессе роботы был разработан алгоритм обработки операторов прерывания и возврата.

Тело выполняемых алгоритмов во внутренней структуре системы имитационного моделирования РДО представляет собой список операций. Каждая операция в таком списке так же может являться списком операций. Благодаря этому в системе возможно строить алгоритмы неограниченной сложности.

Оператор прерывания должен обеспечить выход из выполняемого в данный момент цикла. Оператор возврата должен обеспечивать полный выход из функции и возвращение указанного значения, либо результат вычисления указанного арифметического выражения. При этом оператор прерывания должен обеспечить выход только из текущего цикла, после чего должны продолжить выполнятся оставшиеся после цикла операции, если они присутствуют.

В рамках данной задачи было рассмотрено два варианта обработки операторов прерывания и возврата. Оба варианта будут рассмотрены на тестовом примере. Ниже приведена структура вложенных списков операций использовавшихся для тестирования (Рис. 9.1).

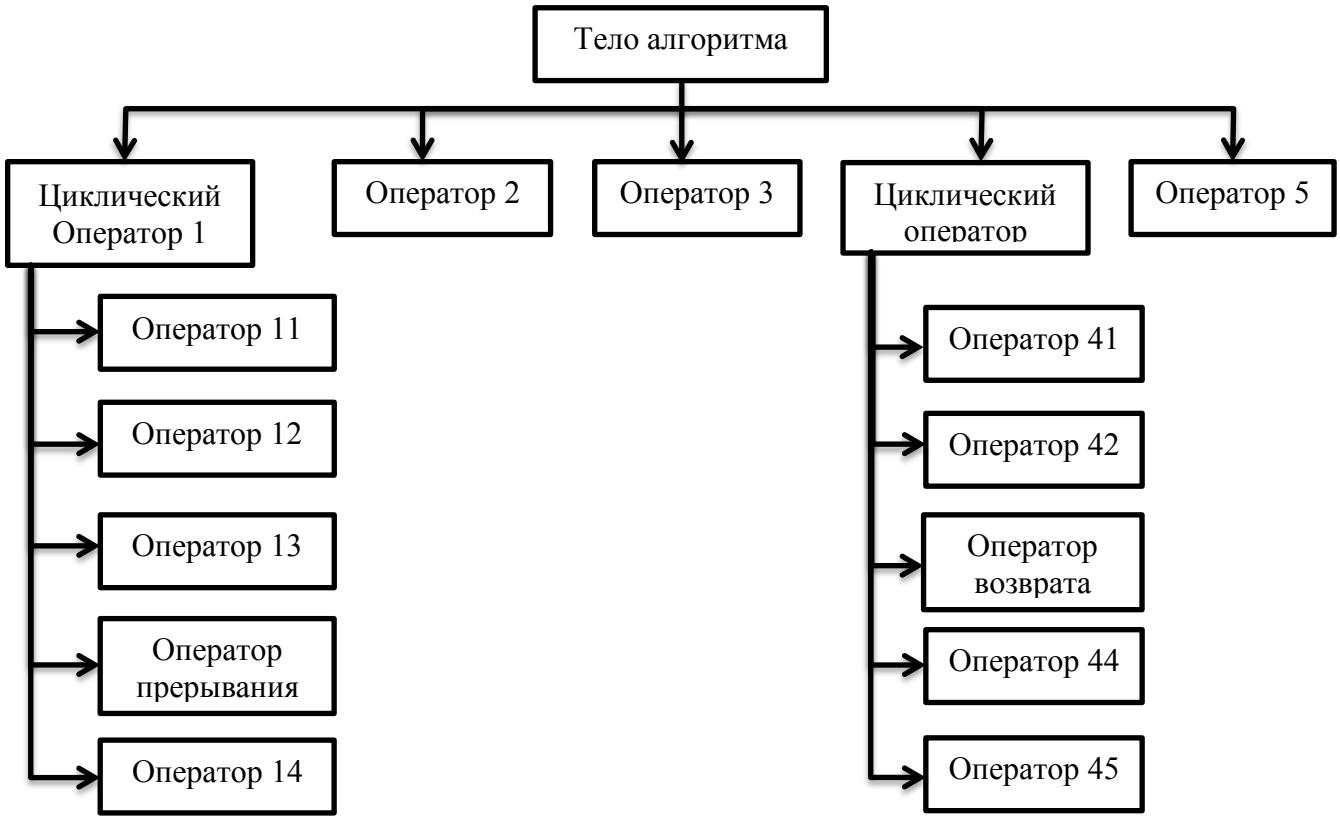

Рис. 9.1 Структура вложенных списков

При работе данного алгоритма оператор выводит на экран свай номер, оператор прерывания выводит текст «break», оператор возврата выводит текст «return». Результат работы будет выглядеть так:

```
11
12
13
break
2
3
41
42
return
```

Первый вариант реализации подразумевает использования обработчика исключений языка C++. Логика реализации данного варианта представлена на диаграмме активности (Рис. 9.2).

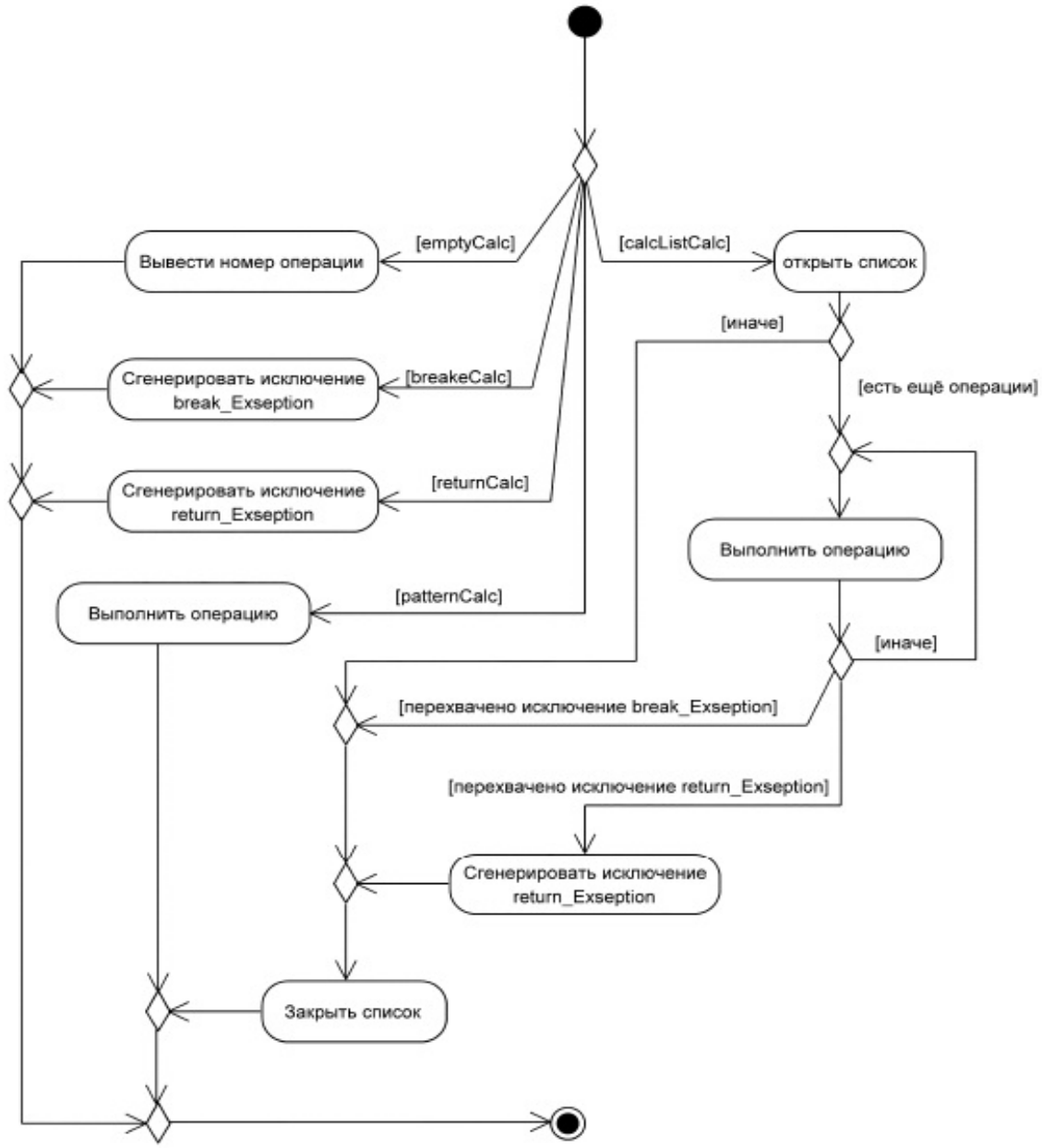

Рис. 9.2 Диаграмма активности. Обработка исключений.

Второй вариант реализации при своей работе использует флаг, который принимает три значения: S_CONTINUE, S_BREAK, S_RETURN. Логика реализации данного варианта представлена на диаграмме активности (Рис. 9.3)

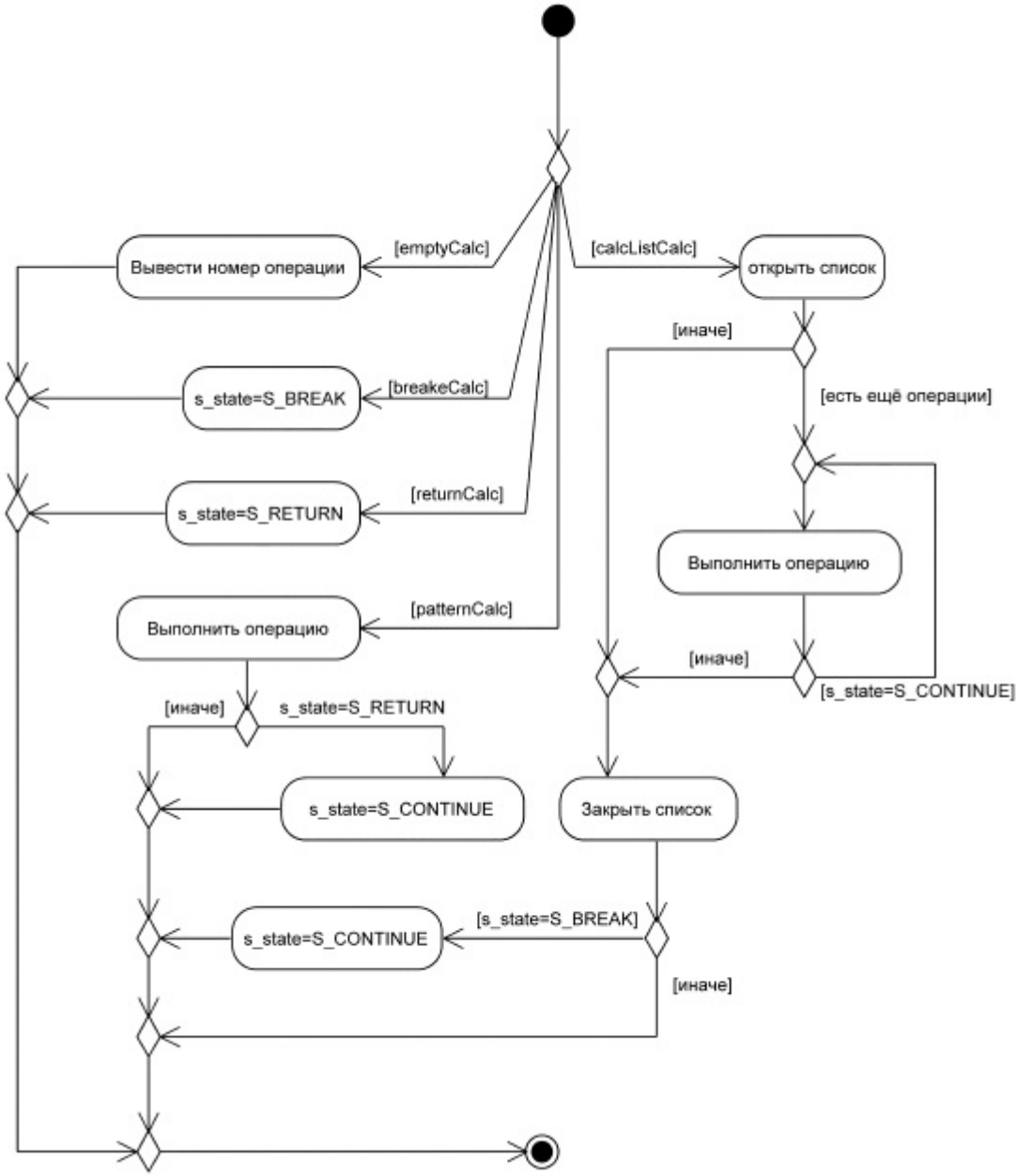

Рис. 9.3 Диаграмма активности. Использование флага.

Для каждого варианта реализации была написана тестовая программа, текст программ приведён в приложении. Было произведено по 10 измерений для разного количества итераций, от 10000 до 300000 с шагом в 10000 итераций. По результатам измерений быстродействия был выбран второй вариант реализации обработки операторов прерывания и возврата. Время работы тестовой программы при реализации первого варианта превосходило аналогичные показатели второго варианта в среднем на **два порядка**. Данное соотношение можно увидеть на графике (Рис. 9.4).

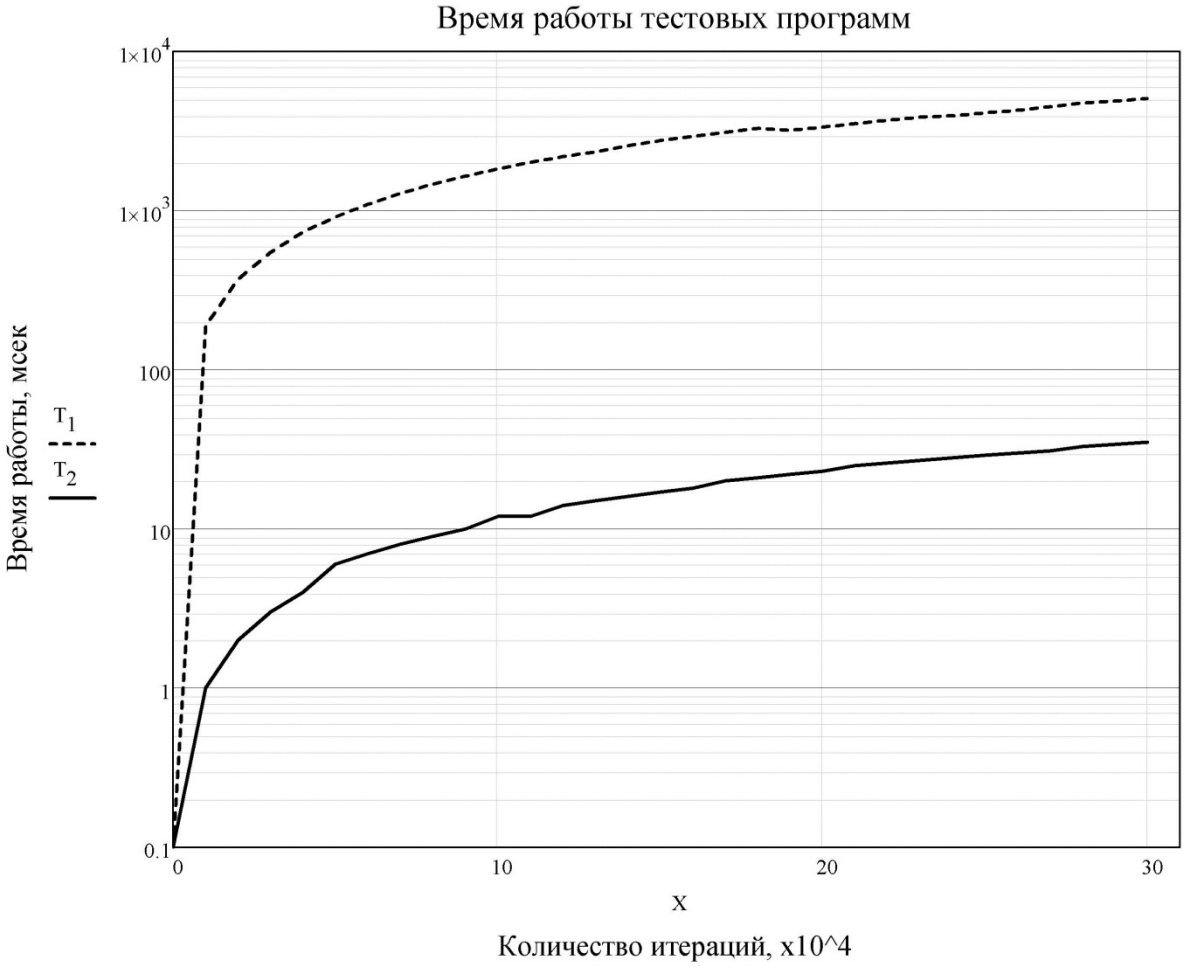

Рис. 9.4 Время работы тестовых программ.

T_1 – Время работы первого варианта реализации (на основе исключений).

T_2 – Время работы второго варианта реализации (на основе флага).

10. ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

10.1. Введение

Современная инженерная деятельность предполагает не только разработку современных конструкций и технологий, но также и концентрацию усилий специалиста, позволяющую заранее определить возможный рынок реализации разработки, оценить ожидаемую прибыль. Поэтому важной составляющей любого инженерного проекта является раздел, посвященной анализу экономических характеристик и определению экономических параметров, позволяющих сделать вывод о возможности реализации инженерной мысли.

В этой главе будет выполнена организационно-экономическая часть дипломного проекта, будет произведён расчет затрат на разработку процедурного языка программирования в системе имитационного моделирования РДО. Для этого будут произведены следующие расчёты:

- Оценка трудоёмкости разработки и внедрения программного обеспечения;
- Оценка состава и численности разработчиков программного обеспечения;
- Оценка трудоёмкости работ каждого участника проектной группы;
- Оценка стоимости разработки процедурного языка программирования в системе имитационного моделирования РДО;

Расчет действителен на 2-ой квартал 2011 года (цены на программное обеспечение, оборудование, расходные материалы, уровень заработной платы исполнителей и т.д.).

10.2. Организация и планирование процесса разработки ПП

В данном разделе дипломного проекта, посвященном экономическим вопросам, будет произведен расчет трудоемкости и стоимости разработки ПП. Целью данного расчета является - определение затрат на разработку ПП.

Организация и планирование процесса разработки ПП предусматривает выполнение следующих работ:

- формирование состава выполняемых работ и группировка их по стадиям разработки;
- расчет трудоемкости выполнения работ;
- установление профессионального состава и расчет количества исполнителей;
- определение продолжительности выполнения отдельных этапов разработки;
- построение календарного графика выполнения разработки;
- контроль выполнения календарного графика.

Трудоемкость разработки ПП зависит от ряда факторов, рассмотрим эти факторы применительно к данной системе.

По степени новизны разрабатываемый программный продукт может быть отнесен к группе новизны «Б» (разработка ПП, не имеющей аналогов, в том числе разработка ППП).

По степени сложности алгоритма функционирования - к 1 группе сложности (реализующие оптимизационные и моделирующие алгоритмы)

По виду представления исходной информации ПП относится к группе 11 – исходная информация представлена в форме документов, имеющих различный формат и структуру. Требуется учитывать взаимовлияние показателей в различных документах.

По структуре выходной информации относим ПП к группе 21 - требуется вывод на печать документов многоуровневой структуры.

10.2.1. Формирование состава выполняемых работ и группировка их по стадиям разработки

Стадия разработки программного продукта	Состав выполняемых работ
Техническое задание	Постановка задач, выбор критериев эффективности. Разработка технико-экономического обоснования разработки. Определение состава пакета прикладных программ, состава и структуры информационной базы. Предварительный выбор методов выполнения работы. Разработка календарного плана выполнения работ.
Эскизный проект	Предварительная разработка структуры входных и выходных данных. Разработка общего описания алгоритмов реализации решения задач. Разработка пояснительной записки. Консультации разработчиков постановки задач. Согласование и утверждение эскизного проекта.
Технический проект	Разработка алгоритмов решения задач. Разработка пояснительной записки. Согласование и утверждение технического проекта. Разработка структуры программы. Разработка программной документации и передача ее для включения в технический проект. Уточнение структуры, анализ и определение формы представления входных и выходных данных. Выбор конфигурации технических средств.
Рабочий проект	Комплексная отладка задач и сдача в опытную эксплуатацию. Разработка проектной документации. Программирование и отладка программ. Описание контрольного примера. Разработка программной документации. Разработка, согласование программы и методики испытаний. Предварительное проведение всех видов испытаний.
Внедрение	Подготовка и передача программной документации для сопровождения с оформлением соответствующего Акта приема-сдачи работ. Проверка алгоритмов и программ решения задач, корректировка документации после опытной эксплуатации программного продукта.

10.2.2. Расчет трудоемкости выполнения работ

$t_{пп}$ - трудоемкость разработки ПП определяется, как сумма величин трудоемкости выполнения отдельных стадий разработки. Формула для определения трудоемкость разработки ПП имеет вид:

$t_{ПП} = t_{ТЗ,} + t_{ЭП} + t_{ТП} + t_{РП} + t_{В}$, где:

$t_{ТЗ,}$ - трудоемкость разработки технического задания на ПП, чел.-дни;

$t_{ЭП}$ - трудоемкость разработки эскизного проекта ПП, чел.-дни;

$t_{ТП}$ - трудоемкость разработки технического проекта ПП, чел.-дни;

$t_{РП}$ - трудоемкость разработки рабочего проекта ПП, чел.-дни;

$t_{В}$ - трудоемкость внедрения разработанного ПП, чел.-дни.

10.2.2.1. Расчет трудоемкости выполнения технического задания

Трудоемкость разработки технического задания рассчитывается по формуле:

$t_{ТЗ,} = t_{РЗ,} + t_{РП}$, где:

$t_{РЗ,}$ - затраты времени разработчика постановки задачи на разработку ТЗ, чел.-дни;

$t_{РП}$ - затраты времени разработчика ПП на разработку ТЗ, чел.-дни.

$t_{РЗ,} = t_{З} \cdot K_{РЗ}$

$t_{РП} = t_{З} \cdot K_{РП}$

где:

$t_{З}$ - норма времени на разработку ТЗ на ПП, чел.-дн.

Исходя из рекомендаций в данном случае $t_{З} = 10$ чел.-дн. (Группа новизны ПП -Б; функциональное назначение ПП – реализующие оптимизационные и моделирующие алгоритмы);

$K_{РЗ}$ - коэффициент, учитывающий удельный вес трудоёмкости работ, выполняемых разработчиком постановки задачи на стадии ТЗ. В случае совместной с разработчиком ПО разработки: $K_{РЗ} = 0,65$;

$K_{РП}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком ПП на стадии ТЗ. В случае совместной с разработчиком ПО разработки: $K_{РП} = 0,35$.

$t_{ТЗ} = 10 \cdot 0,65 + 10 \cdot 0,35 = 10$ чел.-дней.

10.2.2.2. Расчет трудоемкости выполнения эскизного проекта

Трудоемкость разработки эскизного проекта $t_{ЭП}$ рассчитывают по формуле:

$t_{ЭП} = t_{РЗ} + t_{РП}$, где:

$t_{РЗ}$ - затраты времени разработчика постановки задач на разработку ЭП, чел.-дни;

$t_{РП}$ - затраты времени разработчика ПП на разработку ЭП, чел.-дни;

$t_{РЗ} = t_{Э} \cdot K_{РЗ}$,

$t_{РП} = t_{Э} \cdot K_{РП}$

где:

$t_{Э}$ - норма времени на разработку ЭП программного продукта, чел.-дни.

$t_{Э} = 117$ чел.-дней;

$K_{РЗ}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ЭП . В случае совместной с разработчиком ПО разработки ЭП: $K_{РЗ} = 0,7$;

$K_{РП}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком ПП на стадии ЭП. В случае совместной с разработчиком постановки задач разработки ЭП: $K_{РП} = 0,3$;

$t_{ЭП} = 117 \cdot 0,7 + 117 \cdot 0,3 = 117$ чел.-дней.

10.2.2.3. Расчет трудоемкости выполнения технического проекта

Трудоемкость разработки технического проекта зависит от функционального назначения ПП, количества разновидностей форм входной и выходной информации и определяется как сумма времени, затраченного разработчиком постановки задач и разработчиком ПП, и определяется по формуле: $t_{ТП} = \left(t_{РЗ} + t_{РП}\right) \cdot K_{В} \cdot K_{Р}$, где:

$t_{РЗ}$, $t_{РП}$ - норма времени, затрачиваемого на разработку ТП разработчиком постановки задач и разработчиком ПО:

$t_{РЗ} = 86$ ч. ,

$t_{РП} = 23$ ч.

$K_{Р}$ - коэффициент учёта режима обработки информации:

$K_{Р} = 1.45$

$K_{В}$ - коэффициент учёта вида используемой информации, определяется из выражения:

$$K_В = \frac{K_П \cdot N_П + K_{НС} \cdot N_{НС} + K_Б \cdot N_Б}{N_П + N_{НС} + N_Б},$$

где:

$K_П$, $K_{НС}$, $K_Б$ - значения коэффициентов учёта вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно;

$N_П$, $N_{НС}$, $N_Б$ - количество наборов данных для переменной, нормативно-справочной информации и баз данных соответственно.

Для группы новизны Б:

$K_П = 1,2$ $N_П = 7$

$K_{НС} = 1,08$ $N_{НС} = 1$

$K_Б = 3,12$ $N_Б = 2$

Таким образом $K_В = \frac{1,2 \cdot 2 + 1,08 \cdot 1 + 3,12 \cdot 2}{2 + 1 + 2} = 1,944$

$t_{ТП} = (86 + 23) \cdot 1,944 \cdot 1,45 = 307,25$ чел.-дней

10.2.2.4. Расчет трудоемкости выполнения рабочего проекта

Трудоёмкость разработки рабочего проекта зависит от функционального назначения ПП, количества разновидностей форм входной и выходной информации, сложности алгоритма функционирования, сложности контроля информации, степени использования готовых программных модулей, уровня алгоритмического языка программирования и определяется по формуле:

$$t_{РП} = K_К \cdot K_Р \cdot K_Я \cdot K_З \cdot K_{ИА} \cdot (t_{РЗ} + t_{РП}),$$

где:

$K_Р$ - коэффициент учёта режима обработки информации = 1.45;

$K_К$ - коэффициент учёта сложности контроля информации = 1,16 ;

$K_Я$ - коэффициент учёта уровня алгоритмического языка программирования =1,0 интерпритаторы);

$K_З$ - коэффициент учёта степени использования готовых программных модулей – 0.8; (на 20-25% использования готовых прогр. Модулей);

$K_{ИА}$ - коэффициент учёта вида используемой информации и сложности алгоритма ПП.

Значение коэффициента $K_{ИА}$ определяют из выражения:

$$K_{И\,А} = \frac{K_{П}^{/} \cdot N_{П} + K_{НС}^{/} \cdot N_{НС} + K_{Б}^{/} \cdot N_{Б}}{N_{П} + N_{НС} + N_{Б}},$$

где:

$K_{П}^{/}, K_{НС}^{/}, K_{Б}^{/}$ - значения коэффициентов учёта сложности алгоритма ПП и вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно.

В нашем случае присутствует только один вид используемой информации – переменной; соответствующее группе новизны Б значение коэффициента $K_{П}^{/} = 1.62$

$K_{НС} = 0,97\ K_{Б}^{/} = 0.81$

(группа новизны - Б)

$$K_{И\,А} = \frac{1,62 \cdot 2 + 0,97 \cdot 1 + 0,81 \cdot 2}{2 + 1 + 2} = 1,166$$

$t_{РЗ}$, $t_{РП}$ - норма времени, затраченного на разработку РП на алгоритмическом языке высокого уровня разработчиком постановки задач и разработчиком ПП.

$t_{РЗ} = 28$

$t_{РП} = 156$

$t_{РП} = 1,16 \cdot 1,45 \cdot 1,0 \cdot 0,8 \cdot 1,166 \cdot (28 + 156) = 288,69$ чел.-дней.

10.2.2.5. Расчет трудоемкости выполнения внедрения

Трудоёмкость выполнения стадии внедрения может быть рассчитана по формуле:

$$t_{В} = \left(t_{РЗ} + t_{РП}\right) \cdot K_{К} \cdot K_{Р} \cdot K_{З},$$

где:

$t_{РЗ}$, $t_{РП}$ - норма времени, затраченного разработчиком постановки задач и разработчиком ПО на выполнение процедур внедрения ПП.

$t_{РЗ} = 29$

$t_{РП} = 33$

$t_{В} = (29 + 33) \cdot 1,16 \cdot 1.45 \cdot 0.8 = 83,43$ чел.-дней.

10.2.2.6. Расчет суммарной трудоемкости

Таким образом, суммарная трудоемкость разработки программной продукции:

$t_{ПП} = 57 + 117 + 307,25 + 288,69 + 83,43 = 853,37$ чел.-дней.

Трудоемкость этапов разработки программного продукта (таблица 1):

Таблица 1.

№ пп	Стадия разработки	Трудоемкость чел.-дни
1	Техническое задание	10
2	Эскизный проект	117
3	Технический проект	307,25
4	Рабочий проект	288,69
5	Внедрение	83,43
Итого:		806,37

Для целей контроля и планирования выполнения работ в данном случае используем ленточный график, потому что разработку осуществляет небольшой, стабильный по составу коллектив исполнителей . Для построения ленточного графика необходимо знать срок начала работ, срок окончания работ и количество работников, участвующих на каждом этапе разработки.

Продолжительность выполнения всех работ по этапам разработки программного продукта определяется из выражения

$$T_i = \frac{\tau_i + Q}{n_i}$$, где

τ_i - трудоемкость i-й работы, чел.-дни;

Q - трудоемкость дополнительных работ, выполняемых исполнителем, чел.-дни;

n_i - количество исполнителей, выполняющих i-ю работу, чел.

Пусть разработка системы на стадии разработки технического задания ведется одним специалистом, не привлекаемым к дополнительным работам, на стадии разработки эскизного проекта – тремя специалистами, на стадии разработки технического и рабочего проекта - четырьмя специалистами, а на стадии внедрения – двумя специалистами, то продолжительность разработки программного продукта:

$$T = \frac{10}{1} + \frac{117}{3} + \frac{307,25}{4} + \frac{288,69}{4} + \frac{83,43}{2} = 239,7 \text{ раб.дня.}$$

Для построения календарного плана необходимо перевести рабочие дни в календарные. Для этого длительность каждого этапа нужно разделить на поправочный коэффициент К=0.7. В результате получим:

$$T = \frac{239,7}{0.7} = 342,43 \text{ кал.дн} \qquad T_{ТЗ} = \frac{10}{0.7} = 14,29 \text{ кал.дн}$$

$$Tэп = \frac{117}{0,7 \cdot 3} = 55,71 \text{ кал.дн} \qquad T_{ТП} = \frac{307,25}{0,7 \cdot 4} = 109,73 \text{ кал.дн}$$

$$T_{РП} = \frac{288,69}{0,7 \cdot 4} = 103,10 \text{ кал.дн} \qquad Tв = \frac{83,43}{0,7 \cdot 2} = 59,59 \text{ кал.дн}$$

10.3. Определение стоимости разработки ПП

Для определения стоимости работ, необходимо на основании плановых сроков выполнения работ и численности исполнителей, рассчитать общую сумму затрат на разработку программного продукта.

Себестоимость продукции (работ, услуг) представляет собой стоимостную оценку используемых в процессе производства продукции работ, услуг, природных ресурсов, сырья материалов, топлива, энергии, основных фондов, трудовых ресурсов, а также других затрат на ее производство и реализацию.

Затраты, образующие себестоимость продукции (работ, услуг), группируются в соответствии с их экономическим содержанием по следующим элементам:

- расчёт основной заработной платы;
- Расчёт дополнительной заработной платы;
- Отчисления на социальное страхование;
- Накладные расходы;
- Расходы на использование оборудования.

10.3.1. Расчёт основной заработной платы

В статью включается основная заработная плата всех исполнителей, непосредственно занятых разработкой данного ПП, с учётом их должностного оклада и времени участия в разработке. Расчёт ведётся по формуле: $C_{зо} = \sum_i \frac{З_i}{d} \cdot \tau_i$,

где $З_i$ - среднемесячный оклад i-го исполнителя, руб.; d – среднее количество рабочих дней в месяце; τ_i - трудоемкость работ, выполняемых i-м исполнителем, чел.-дни (определяется из календарного плана-графика).

$З_i = 30000\,руб.$;

$d = 21$;

$\tau_i = 853,37$

$C_{зо} = \frac{30000}{21} \cdot 853,37 = 1219100\,руб.$

10.3.2. Расчёт дополнительной заработной платы

В статье учитываются все выплаты непосредственным исполнителям за время (установленное законодательством), непроработанное на производстве, в том числе: оплата очередных отпусков, компенсация за неиспользованный отпуск, оплата льготных часов подросткам и др. Расчет ведётся по формуле: $C_{зд} = C_{зо} \cdot \alpha_д$,

где $\alpha_д$ - коэффициент на дополнительную заработную плату.

$\alpha_д = 0,2$

$C_{зд} = 1219100 \cdot 0,2 = 243820\,руб.$

10.3.3. Отчисления на социальное страхование

В статье учитываются отчисления в бюджет социального страхования по установленному законодательному тарифу от суммы основной и дополнительной заработной платы, т.е.

$C_{сс} = \alpha_{сс} \cdot (C_{зо} + C_{зд})$

$\alpha_д = 0,34 + 0,02 = 0,36$

$C_{зд} = (1219100 + 243820) \cdot 0,36 = 526651,2\,руб.$

10.3.4. Накладные расходы

В статье учитываются затраты на общехозяйственные расходы, непроизводительные расходы и расходы на управление. Накладные расходы определяют в процентном отношении к основной заработной плате, т.е.

$C_н = C_{зо} \cdot \alpha_н,$

$\alpha_н = 1,8$

$C_{зд} = 1219100 \cdot 1,8 = 2194380\ руб.$

10.3.5. Расходы на использование оборудования

Затраты, связанные с использованием вычислительной техники, определяются по формуле: $C_{ЭВМ} = t^{ЭВМ} \cdot K_И^{ЭВМ} \cdot Ц^{ЭВМ} \cdot K_{БД}^{ЭВМ} \cdot K_Э^{ЭВМ}$, где

$t^{ЭВМ}$ -время использования ЭВМ для разработки данного ПП, ч;

$K_И^{ЭВМ}$ -поправочный коэффициент учета времени использования ЭВМ;

$Ц^{ЭВМ}$ - цена I-го часа работы ЭВМ;

$K_{БД}^{ЭВМ}$ - коэффициент учета степени использования СУБД;

$K_Э^{ЭВМ}$ - коэффициент учета быстродействия ЭВМ.

$t^{ЭВМ} = 0,9 \cdot 806,37 \cdot 8 = 5805,86ч.$ $K_И^{ЭВМ} = 1,26$

$Ц^{ЭВМ} = 35\ руб.$ $K_{БД}^{ЭВМ} = 1,0$ $K_Э^{ЭВМ} = 1,0$

$C_{ЭВМ} = 5805,86 \cdot 1,26 \cdot 35 \cdot 1,0 \cdot 1,0 = 256038,43\ руб.$

10.3.6. Результаты расчетов затрат на разработку программного продукта

№ пп	Наименование статьи	Сметная стоимость, руб.
1	основная заработная плата	1219100
2	Дополнительная заработная плата	243820
3	Отчисления на социальное страхование	526651,2
4	Накладные расходы	2194380
5	Расходы на использование оборудования	256038,43
	Итого	4439989,63

10.4. Выводы к главе

Произведенный расчет показал:

-Суммарная трудоемкость продукта составляет 806,37 чел.дней. Поэтому на каждом этапе необходимо привлечение нескольких специалистов.

-Длительность разработки программного продукта составляет 342,43 календарных дней

-Себестоимость программного продукта составляет 4215955,89 руб.

11. ЭКОЛОГИЧЕСКАЯ ЧАСТЬ

11.1. Опасные и вредные факторы

11.1.1. Опасные факторы

- пожарная опасность, обусловленная наличием на рабочем месте мощного источника энергии.
- повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;

11.1.2. Вредные факторы

11.1.2.1. Физические

- повышенные уровни электромагнитного излучения;
- повышенные уровни рентгеновского излучения;
- повышенные уровни ультрафиолетового излучения;
- повышенный уровень инфракрасного излучения;
- повышенный уровень статического электричества;
- повышенные уровни запыленности воздуха рабочей зоны;
- повышенное содержание положительных аэроионов в воздухе рабочей зоны;
- пониженное содержание отрицательных аэроионов в воздухе рабочей зоны;
- пониженная или повышенная влажность воздуха рабочей зоны;
- пониженная или повышенная подвижность воздуха рабочей зоны;
- повышенный уровень шума;
- повышенный или пониженный уровень освещенности;
- повышенный уровень прямой блесткости;
- повышенный уровень отраженной блесткости;
- повышенный уровень ослепленности;
- неравномерность распределения яркости в поле зрения;
- повышенная яркость светового изображения;
- повышенный уровень пульсации светового потока;

11.1.2.2. Химические

- повышенное содержание в воздухе рабочей зоны двуокиси углерода, озона, аммиака, фенола, формальдегида и полихлорированных бифенилов;

11.1.2.3. Психофизиологические

- напряжение зрения;
- напряжение внимания;
- интеллектуальные нагрузки;
- эмоциональные нагрузки;
- длительные статические нагрузки;
- монотонность труда;
- большой объем информации обрабатываемой в единицу времени;
- нерациональная организация рабочего места;

11.1.2.4. Биологические

- повышенное содержание в воздухе рабочей зоны микроорганизмов.

11.2. Охрана труда

11.2.1. Введение

Усложнение производственной деятельности в связи с применением ПЭВМ предъявляют новые требования к организму человека. Неправильное или невнимательное отношение к выполнению требований техники безопасности при работе на ПЭВМ неизбежно отражается на показателях деятельности и здоровье работников. Рабочее место (РМ) оператора ПЭВМ представляет собой совокупность физических, химических, биологических, социально-психологических и эстетических факторов внешней среды, воздействующих на оператора. Основные требования к факторам рабочей среды заключаются в следующем:

- факторы рабочей среды при их комплексном воздействии на человека не должны оказывать отрицательного влияния на его здоровье при профессиональной деятельности его в течение длительного времени;
- факторы рабочей среды не должны вызывать снижения надежности и качества деятельности оператора при действии их в течение рабочего дня.

11.2.2. Требования к ПЭВМ

ПЭВМ должны соответствовать требованиям СанПиН 2.2.2/2.4.1340-03, и каждый их тип подлежит санитарно-эпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке.

Перечень продукции и контролируемых гигиенических параметров вредных и опасных факторов представлен в приложении 1 (таблица 1).

Допустимые уровни звукового давления и уровней звука, создаваемого ПЭВМ, не должны превышать значений, представленных в приложении 1 (таблица 2).

Временные допустимые уровни электромагнитных полей (ЭМП), создаваемых ПЭВМ, не должны превышать значений, представленных в приложении 1 (таблица 3).

Допустимые визуальные параметры устройств отображения информации представлены в приложении 1 (таблица 4).

Концентрации вредных веществ, выделяемых ПЭВМ в воздух помещений, не должны превышать предельно допустимых концентраций (ПДК), установленных для атмосферного воздуха ("Предельно допустимые концентрации загрязняющих веществ в атмосферном воздухе населенных мест". ГН 2.1.6.695-98 (ПДК среднесуточная: окись углерода – 3,0 мг/м^3, окислы азота – 0,04 мг/м^3)).

Мощность экспозиционной дозы мягкого рентгеновского излучения в любой точке на расстоянии 0,05 м от экрана и корпуса ВДТ (на электронно-лучевой трубке) при любых положениях регулировочных устройств не должна превышать 1 мкЗв/час (100 мкР/час).

Конструкция ПЭВМ должна обеспечивать возможность поворота корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана ВДТ. Дизайн ПЭВМ должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света. Корпус ПЭВМ, клавиатура и другие блоки и устройства ПЭВМ должны иметь матовую поверхность с коэффициентом отражения 0,4 - 0,6 и не иметь блестящих деталей, способных создавать блики.

Конструкция ВДТ должна предусматривать регулирование яркости и контрастности.

Документация на проектирование, изготовление и эксплуатацию ПЭВМ не должна противоречить требованиям СанПиН 2.2.2/2.4.1340-03.

11.2.3. Требования к помещениям для работы с ПЭВМ

Помещения для эксплуатации ПЭВМ должны иметь естественное и искусственное освещение. Эксплуатация ПЭВМ в помещениях без естественного освещения допускается только при соответствующем обосновании и наличии положительного санитарно-эпидемиологического заключения, выданного в установленном порядке.

Естественное и искусственное освещение должно соответствовать требованиям действующей нормативной документации. Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток. Оконные проемы должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

Не допускается размещение мест пользователей ПЭВМ в цокольных и подвальных помещениях.

Площадь на одно рабочее место пользователей ПЭВМ с ВДТ на базе электронно-лучевой трубки (ЭЛТ) должна составлять не менее 6 $м^2$ и с ВДТ на базе плоских дискретных экранов (жидкокристаллические, плазменные) - 4,5 $м^2$. При использовании ПВЭМ с ВДТ на базе ЭЛТ (без вспомогательных устройств - принтер, сканер и др.), отвечающих требованиям международных стандартов безопасности компьютеров, с продолжительностью работы менее 4-х часов в день допускается минимальная площадь 4,5 $м^2$ на одно рабочее место пользователя.

Для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно отражающие материалы с коэффициентом отражения для потолка - 0,7 - 0,8; для стен - 0,5 - 0,6; для пола - 0,3 - 0,5.

Полимерные материалы используются для внутренней отделки интерьера помещений с ПЭВМ при наличии санитарно-эпидемиологического заключения.

Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

11.2.4. Требования к микроклимату, содержанию аэроионов и вредных химических веществ в воздухе на рабочих местах, оборудованных ПЭВМ

В помещениях, где расположены ПЭВМ, должны обеспечиваться оптимальные параметры микроклимата (приложение 2, таблица 3). Данный микроклимат может обеспечиваться системой общеобменной приточно-вытяжной механической вентиляции, которая должна обеспечить объём подаваемого воздуха в единицу времени не менее 20 $м^3$/час и температуру подаваемого воздуха не ниже 10° Цельсия. В приточной ветви нужно предусмотреть использование фильтров очистки воздуха, для удаления из воздуха вредных веществ перед его подачей на рабочие места. Для повышения влажности воздуха в помещениях с ВДТ и ПЭВМ следует применять увлажнители воздуха, заправляемые ежедневно дистиллированной или прокипяченной питьевой водой.

В помещениях, оборудованных ПЭВМ, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПЭВМ.

Уровни положительных и отрицательных аэроионов в воздухе помещений, где расположены ПЭВМ, должны соответствовать действующим санитарно-эпидемиологическим нормативам:

Уровни	Число ионов в 1 см. куб. воздуха	
	n+	n-
Минимально необходимые	400	600
Оптимальные	1500-3000	3000-5000
Максимально допустимые	50000	50000

Содержание вредных химических веществ в воздухе помещений, предназначенных для использования ПЭВМ не должно превышать предельно допустимых среднесуточных концентраций для атмосферного воздуха в соответствии с действующими санитарно-эпидемиологическими нормативами ("Предельно допустимые концентрации загрязняющих веществ в атмосферном воздухе населенных мест". ГН 2.1.6.695-98 (ПДК среднесуточная: окись углерода – 3,0 $мг/м^3$, окислы азота – 0,04 $мг/м^3$)).

11.2.5. Требования к уровням шума и вибрации на рабочих местах, оборудованных ПЭВМ

В помещениях, где расположены ПЭВМ, уровни шума не должны превышать допустимых значений, установленных для жилых и общественных зданий ("Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки" СН 2.2.4/2.1.8.562-96):

Уровни звукового давления, дБ, в октавных полосах со среднегеометрическими частотами, Гц									Уровни звука L_A и эквивалентные уровни звука $L_{Аэкв}$, дБА	Максимальные уровни звука $L_{Амакс}$, дБА
1,5	3	25	50	00	000	000	000	000		
9	3	2	5	9	5	2	0	8	40	55

Снизить уровень шума в помещениях с ВДТ и ПЭВМ можно использованием звукопоглощающих материалов с максимальными коэффициентами звукопоглощения в области частот 63 - 8000 Гц. Дополнительным звукопоглощением служат однотонные занавеси из плотной ткани, гармонирующие с окраской стен и подвешенные в складку на расстоянии 15 - 20 см от ограждения. Ширина занавеси должна быть в 2 раза больше ширины окна.

В помещениях, в которых эксплуатируются ПЭВМ, уровень вибрации не должен превышать допустимых значений для жилых и общественных зданий в соответствии с действующими санитарно-эпидемиологическими нормативами ("Производственная вибрация, вибрация в помещениях жилых и общественных зданий" СН 2.2.4/2.1.8.566-96):

Среднегеометрические частоты октавных полос, Гц	Допустимые значения оси X, Y			
	по виброускорению		по виброскорости	
	$м/с^2$ х	д	м/с х	д
2	10	8	0,79	8
4	11	8	0,45	7
8	14	8	0,28	7
16	28	8	0,28	7
31,5	56	9	0,28	7
63	110	1	0,28	7
Корректированные значения и их	10	8	0,28	7

Примечания: 1) Для непостоянной вибрации к допустимым значениям уровней, приведенным в табл.10, вводится поправка - 10 дБ, а абсолютные значения умножаются на 0,32; 2) для помещений учебных заведений вводится поправка - 3 дБ.

Для снижения вибрации в помещениях оборудование, аппараты и приборы необходимо устанавливать на амортизирующие прокладки. Также могут быть использованы средства индивидуальной защиты.

Шумящее оборудование (печатающие устройства, серверы и т.п.), уровни шума которого превышают нормативные, должно размещаться вне помещений с ПЭВМ.

11.2.6. Требования к освещению на рабочих местах, оборудованных ПЭВМ

Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300 - 500 лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк.

Следует ограничивать прямую блесткость от источников освещения, при этом яркость светящихся поверхностей (окна, светильники и др.), находящихся в поле зрения, должна быть не более 200 кд/м^2.

Следует ограничивать отраженную блесткость на рабочих поверхностях (экран, стол, клавиатура и др.) за счет правильного выбора типов светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при этом яркость бликов на экране ПЭВМ не должна превышать 40 кд/м^2 и яркость потолка не должна превышать 200 кд/м^2.

Показатель дискомфорта в помещениях - не более 15.

Яркость светильников общего освещения в зоне углов излучения от 50 до 90 градусов с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м^2, защитный угол светильников должен быть не менее 40 градусов.

Светильники местного освещения должны иметь не просвечивающий отражатель с защитным углом не менее 40 градусов.

Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ, при этом соотношение яркости между рабочими поверхностями не должно превышать 3:1 - 5:1, а между рабочими поверхностями и поверхностями стен и оборудования - 10:1.

В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ). В светильниках местного освещения допускается применение ламп накаливания, в том числе галогенных.

Для освещения помещений с ПЭВМ следует применять светильники с зеркальными параболическими решетками, укомплектованными электронными пускорегулирующими аппаратами (ЭПРА). Допускается использование многоламповых светильников с электромагнитными пускорегулирующими аппаратами (ЭПРА), состоящими из равного числа опережающих и отстающих ветвей.

Применение светильников без рассеивателей и экранирующих решеток не допускается.

При отсутствии светильников с ЭПРА лампы многоламповых светильников или рядом расположенные светильники общего освещения следует включать на разные фазы трехфазной сети.

Общее освещение при использовании люминесцентных светильников следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядном расположении видеодисплейных терминалов. При периметральном расположении компьютеров линии светильников должны располагаться локализовано над рабочим столом ближе к его переднему краю, обращенному к оператору.

Коэффициент запаса (Кз) для осветительных установок общего освещения должен приниматься равным 1,4.

Коэффициент пульсации не должен превышать 5%.

Для обеспечения нормируемых значений освещенности в помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

11.2.7. Требования к уровням электромагнитных полей на рабочих местах, оборудованных ПЭВМ

Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах пользователей, представлены в приложении 2 (таблица 1).

Методика проведения инструментального контроля уровней ЭМП на рабочих местах пользователей ПЭВМ представлена в приложении 3.

11.2.8. Требования к визуальным параметрам ВДТ, контролируемым на рабочих местах

Предельно допустимые значения визуальных параметров ВДТ, контролируемые на рабочих местах, представлены в приложении 2

(таблица 2).

11.2.9. Общие требования к организации рабочих мест пользователей ПЭВМ

При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м.

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

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики. Поверхность рабочего стола должна иметь коэффициент отражения 0,5 - 0,7.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

11.2.10. Требования к организации и оборудованию рабочих мест с ПЭВМ для пользователей

Помещения для занятий оборудуются одноместными столами, предназначенными для работы с ПЭВМ.

Конструкция одноместного стола для работы с ПЭВМ должна предусматривать:

- две раздельные поверхности: одна горизонтальная для размещения ПЭВМ с плавной регулировкой по высоте в пределах 520 – 760 мм и вторая – для клавиатуры с плавной регулировкой по высоте и углу наклона от 0 до 15 градусов с надежной фиксацией в оптимальном положении (12 – 15 градусов);
- ширину поверхностей для ВДТ и клавиатуры не менее 750 мм (ширина обеих поверхностей должна быть одинаковой) и глубину не менее 550 мм;
- опору поверхностей для ПЭВМ или ВДТ и для клавиатуры на стояк, в котором должны находиться провода электропитания и кабель локальной сети. Основание стояка следует совмещать с подставкой для ног;
- отсутствие ящиков;
- увеличение ширины поверхностей до 1200 мм при оснащении рабочего места принтером.

Высота края стола, обращенного к работающему с ПЭВМ и высота пространства для ног должны соответствовать росту обучающихся в обуви (приложение 4).

При наличии высокого стола и стула, несоответствующего росту обучающихся, следует использовать регулируемую по высоте подставку для ног.

Линия взора должна быть перпендикулярна центру экрана и оптимальное ее отклонение от перпендикуляра, проходящего через центр экрана в вертикальной плоскости не должна превышать +-5 градусов, допустимое +- 10 градусов.

Рабочее место с ПЭВМ оборудуют стулом, основные размеры которого должны соответствовать росту обучающихся в обуви (приложение 5).

11.2.11. Требования к организации медицинского обслуживания пользователей ПЭВМ

Лица, работающие с ПЭВМ более 50% рабочего времени (профессионально связанные с эксплуатацией ПЭВМ), должны проходить обязательные предварительные при поступлении на работу и периодические медицинские осмотры в установленном порядке.

Женщины со времени установления беременности переводятся на работы, не связанные с использованием ПЭВМ, или для них ограничивается время работы с ПЭВМ (не более 3-х часов за рабочую смену) при условии соблюдения гигиенических требований, установленных СанПиН 2.2.2/2.4.1340-03. Трудоустройство беременных женщин следует осуществлять в соответствии с законодательством Российской Федерации.

11.2.12. Требования к проведению государственного санитарно-эпидемиологического надзора и производственного контроля

Государственный санитарно-эпидемиологический надзор за производством и эксплуатацией ПЭВМ осуществляется в соответствии с СанПиН 2.2.2/2.4.1340-03.

Не допускается реализация и эксплуатация на территории Российской Федерации типов ПЭВМ, не имеющих санитарно-эпидемиологического заключения.

Инструментальный контроль за соблюдением требований СанПиН 2.2.2/2.4.1340-03 осуществляется в соответствии с действующей нормативной документацией.

Производственный контроль за соблюдением санитарных правил осуществляется производителем и поставщиком ПЭВМ, а также предприятиями и организациями, эксплуатирующими ПЭВМ в установленном порядке, в соответствии с действующими санитарными правилами и другими нормативными документами.

11.2.13. Требования электробезопасности

В соответствии с правилами устройства электрических установок в помещениях без повышенной электрической опасности заземлению подлежат токоприемники с напряжением U≥380В при переменном токе, а в помещениях с повышенной электрической опасностью – токоприемники с напряжением U≥42В при переменном токе. К помещениям с повышенной электрической опасностью относятся помещения с токопроводящими полами или при наличии токопроводящей пыли, с повышенной влажностью (>75%) или жаркие помещения (>35°С), а также помещения, где существует опасность касания токоприемников или их элементов. Если оператор будет находиться в помещении с повышенной электрической опасностью, то ПЭВМ и ВДТ необходимо заземлять (220В>42В), если же помещение будет относиться к категории помещений без повышенной электрической опасности, то ВДТ и ПЭВМ заземлять не надо (220В<380В). Необходимо, также использовать заземленные защитные экраны и фильтры.

11.2.14. Требования пожарной безопасности

Так как в помещении имеется электрооборудование, следовательно, есть опасность возгорания. По классификации пожарной опасности помещение, в котором будет вестись работа с программным комплексом, относится к категории В, так как там находится мебель, как мягкая, содержащая обивку и поролон, так и из ДСП, пластика и дерева. Следовательно, помещение должно быть оборудовано порошковым огнетушителем, планом эвакуации при пожаре и аптечкой первой помощи. В помещении должен быть ответственный за пожарную безопасность.

11.2.15. Расчет системы защитного заземления компьютера

Данное помещение является помещением с повышенной электроопасностью, т.к. имеется возможность касания токоприемников (ВДТ и ПЭВМ) с проводящими системами, связанными с землей (система централизованного отопления), поэтому заземление необходимо.

Определяем расчетное удельное сопротивление грунта $\rho_{расч}$ с учетом климатического коэффициента Ψ (табл. 5 и 6): $\rho_{расч} = \rho_{изм} \cdot \Psi$, где $\rho_{изм}$ - удельное сопротивление грунта, измеренное или полученное из справочной литературы;

Таблица 5. Приближенные значения удельных сопротивлений грунтов

Грунт	Удельное сопротивление, 10^2 Ом·м	
	Возможные пределы колебаний	При влажности 10-12% к массе грунта
Песок	4-7	7
Супесок	1,5-4	3
Суглинок	0,4-1,5	1
Глина	0,08-0,7	0,4
Чернозем	0,09-5,3	2
Речная вода	0,5	-
Морская вода	0,002-0,01	-

Выбираем суглинок, как наиболее близкий тип почвы. При нормальной влажности почвы (10-12%), получаем $\rho_{изм}$ = 100 Ом·м.

Таблица 6. Значения расчетных климатических коэффициентов сопротивления грунта

Грунт	Глубина заложения, м.	Ψ_1	Ψ_2	Ψ_3
Суглинок	0,8—3,8	2,0	1,5	1,4

Примечание. Расчетное сопротивление грунта определяется по значениям:

Ψ_1 – при большой влажности грунта; Ψ_2 – при средней влажности грунта; Ψ_3 – при сухом грунте.

Выбираем средне влажный грунт: Ψ = 1,5. Тогда $\rho_{расч}$ = 100·1,5 = 150 Ом·м.

Теперь выбираем тип заземлителя. Выбираем протяженную полосу в грунте. (см. рис.1)

Таблица 7. Формулы для вычисления сопротивлений одиночных заземлителей растеканию тока

Тип заземлителя	Формула	Дополнительные указания
Протяженный – полоса в грунте	$R_3 = \frac{\rho}{2\pi l} \cdot \ln\frac{2l^2}{bH}$	$\frac{l}{H} \geq 5$

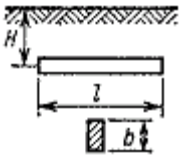

Рис. 11.1 Схема заземлителя.

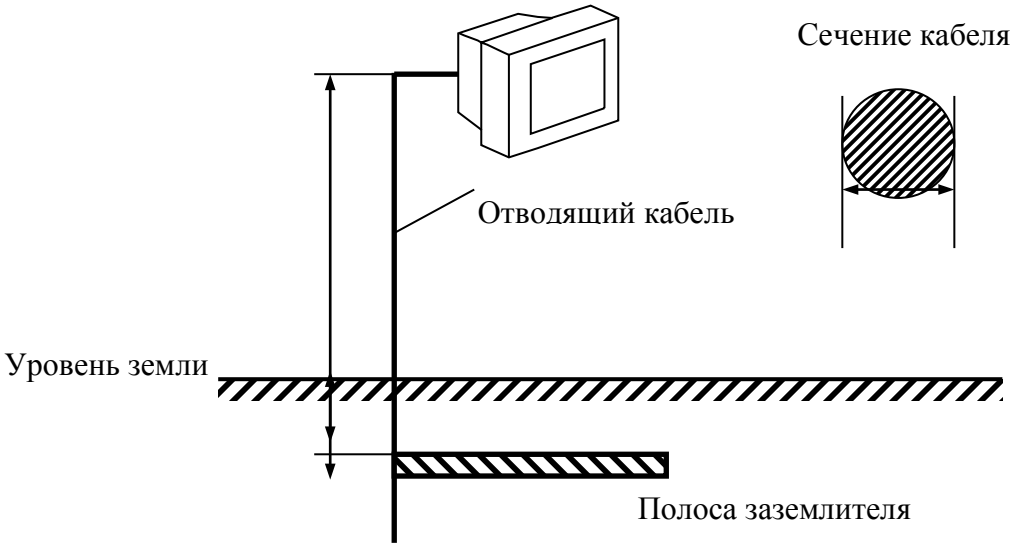

Рис. 11.2 Схема заземления ВДТ.

H = 3 м – глубина залегания полосы заземления ;

h =6 м – высота от уровня земли до ВДТ;

d = 10 мм – диаметр отводящего кабеля.

Материал кабеля: медь.

ρ_{Cu} = 0,0175 Ом·мм^2/м – удельное сопротивление меди.

Сопротивление отводящего кабеля: $R_{ÊÀÁ} = \rho_{Cu} \dfrac{h+H}{\pi \cdot d^2 / 4}$

$R_{ÊÀÁ} = 0.002 \hat{I}\grave{i}$

Общее сопротивление системы заземления: $R_{ОБЩ} = R_З + R_{КАБ}$

ρ = 150 Ом·м;

b = 0,1м.

В целях проведения проверочного расчета принимаем предельный случай $l / H = 5 \Rightarrow l = 15 \grave{i}$.

(Уравнение рассчитано с использованием пакета MathCAD 14)

$$R_Ç = \frac{\rho}{2\pi l} \cdot \ln \frac{2l^2}{bH} \Rightarrow R_Ç = 6.51 \hat{I}\grave{i}$$

$$R_{\hat{I}\acute{A}\grave{U}} = R_{\c{C}} + R_{\hat{E}\grave{A}\acute{A}} \Rightarrow R_{\hat{I}\acute{A}\grave{U}} = 6.512\hat{I}\grave{i} > 4\hat{I}\grave{i}$$

Чтобы напряжение на заземленном корпусе было минимальным, ограничивают сопротивление заземления. Для данного случая с напряжением 380/220В оно должно быть не более 4Ом. Т.е. имеющаяся система защитного заземления компьютера не удовлетворяет требованиям электробезопасности. Необходимо пересмотреть параметры системы защитного заземления, например, варьировать соотношение $\frac{l}{H} \geq 5$

11.3. Утилизация ПЭВМ

Извлечение драгоценных металлов из вторичного сырья является частью проблемы использования возвратных ресурсов, которая включает в себя следующие аспекты: нормативно-правовой, организационный, сертификационный, технологический, экологический, экономико-финансовый. Проблема использования вторичного сырья, содержащего драгоценные материалы из компьютеров, периферийного оборудования и иных средств вычислительной техники (СВТ) актуальна в связи с техническим перевооружением отраслей промышленности.

К драгоценным металлам относятся: золото, серебро, платина, палладий, родий, иридий, рутений, осмий, а также любые химические соединения и сплавы каждого из этих металлов. Статья 2 п. 4 "Федерального закона о драгоценных металлах и драгоценных камнях" от 26 марта 1998 года №1463 гласит: "Лом и отходы драгоценных металлов подлежат сбору во всех организациях, в которых образуются указанные лом и отходы. Собранные лом и отходы подлежат обязательному учёту и могут перерабатываться собирающими их организациями для вторичного использования или реализовываться организациям, имеющим лицензии на данный вид деятельности, для дальнейшего производства и аффинажа драгоценных металлов".

Порядок учёта, хранения, транспортировки, инвентаризации, сбор и сдача отходов драгоценных металлов из СВТ, деталей и узлов, содержащих в своём составе драгоценные металлы для предприятия, учреждения и организации (далее - предприятие), независимо от форм собственности, установлен инструкцией Министерства финансов Российской Федерации от 4 августа 1992 года №67. Все виды работ с драгоценными металлами строго регламентированы нормативно-правовыми документами, перечень которых представлен в Приложении 6.

11.3.1. Разборка изделий

Последовательность разборки определяется типом изделия СВТ, его конструкционными особенностями и комплектацией.

Как правило, процесс разборки должен выполняется в последовательности, обратной процессу сборки изделия. Основные направления деятельности на этапе «Разборка»

11.3.1.1. Разборка персональных компьютеров (ПЭВМ), рабочих станций и серверов

- Технологии разборки ПЭВМ, рабочих станций, серверов и информационно-вычислительных систем едины поскольку состав их модулей стандартный. Он содержит системный блок и комплект периферийных устройств.
- Разборку ПЭВМ и составных модулей целесообразно осуществлять по технологической схеме представленной на рис.7.
- Порядок разборки системного блока.
- Выключить компьютер и отсоединить шнур питания от розетки и системного блока. Отсоединить переходной шнур питания от системного блока к монитору.
- Отсоединить от компьютера клавиатуру, монитор, манипулятор "мышь", принтер, сканер и иные внешние устройства.
- Найти элементы крепления крышки корпуса (винты, шурупы, пружинные защелки и т.д.). Освободить крышку от элемента крепления.
- Снять крышку.
- Отсоединить внутренние кабели и плоские шлейфы.
- Найти элементы крепления дисководов (НМД, НГМД) в отсеке для дисководов (винты, шурупы, саморезные винты, пружинные защелки и др.). Освободить дисководы и извлечь их из дискового отсека.
- Освободить от крепёжных элементов периферийные платы. Извлечь из разъёмов непосредственного контактирования все периферийные платы.
- Найти элементы крепления системной платы к корпусу (винты, шурупы). Освободить элементы крепления и извлечь системную плату из корпуса.

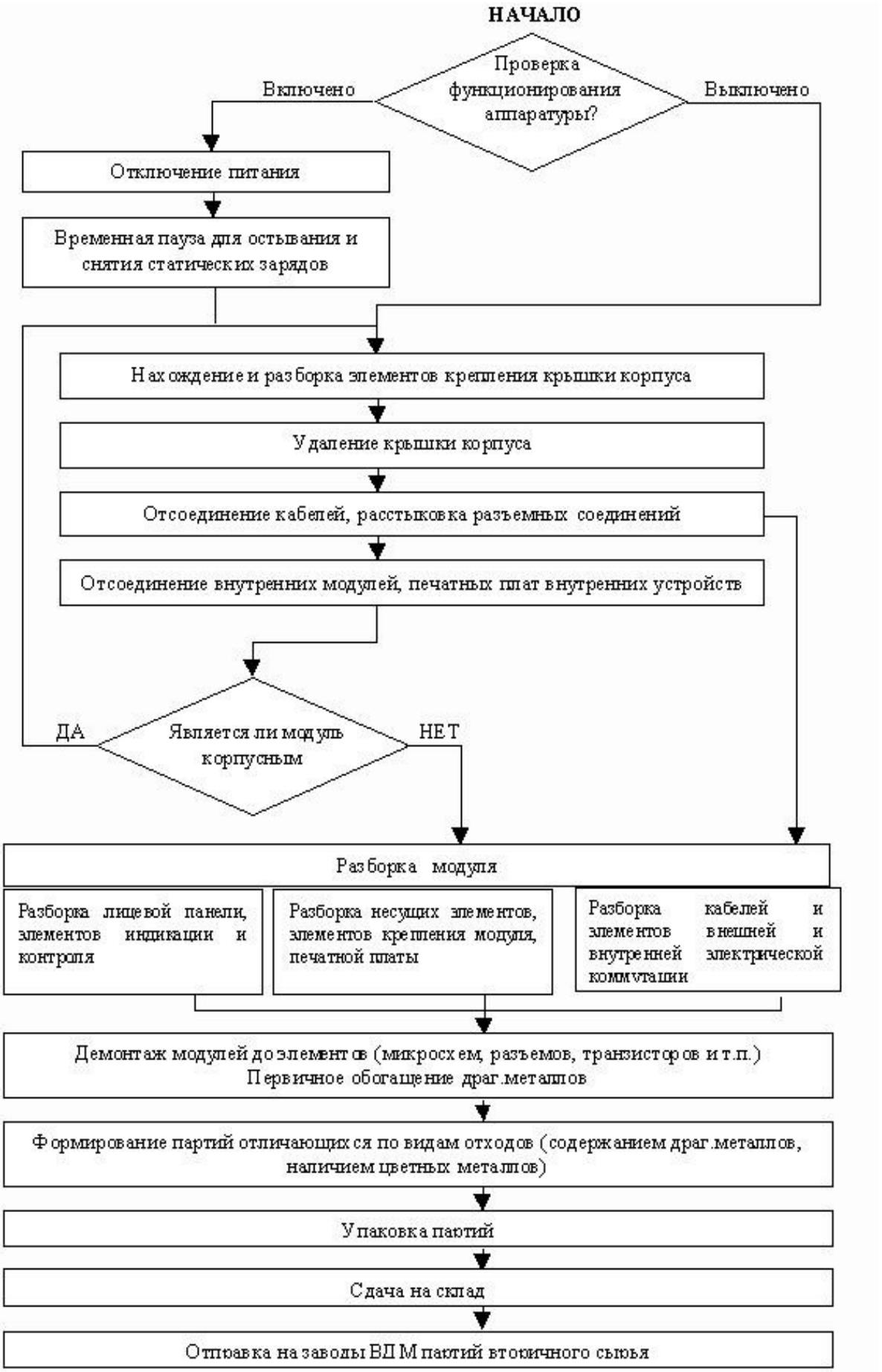

Рис. 11.3 Технологическая схема разборки ПЭВМ.

- Извлечь модули памяти из разъёмов системной платы.
- Найти элементы крепления блока питания к корпусу (винты, шурупы, саморезные винты, пружинные защелки и пр.). Освободить элементы крепления и извлечь блок питания.
- Разобрать блок питания и извлечь высоковольтные конденсаторы содержащие тантал.
- Разобрать ПП и модули памяти до компонентов (микросхем, транзисторов, разъёмов и т.п.).
- Произвести сортировку компонентов и сформировать партии электронного лома.
- Упаковать партии, составить опись, произвести расчёт (анализ) драгметаллов и передать их на склад.
- Провести сортировку цветных и чёрных металлов, пластмасс, сформировать партии и передать их на склад или на переработку.
- При оценке содержания драгоценных металлов в партии электронного лома отечественных ПЭВМ необходимо руководствоваться паспортными данными. При оценке ПЭВМ импортного производства необходимо провести ориентировочные расчёты по отечественным аналогам.

11.3.1.2. Обеспечение комплексности технологии разборки

При разборке изделий СВТ образуются материалы и изделия, которые имеют материальную ценность и подлежат реализации.

Примерный перечень материалов представлен в табл.6.

Таблица 6

Виды материалов и изделий, подлежащих реализации при комплексной переработке СВТ

п.п.	**Вид материалов или изделий**	**Характеристика**
.	Печатные платы, разъемы и соединители, микросхемы	вторичные драгоценные металлы
.	Электрические провода и кабели, соединители	вторичная медь и её сплавы
.	Свинец и олово из печатных плат	вторичные припойные пасты (олово и свинец)

.	Танталовые конденсаторы К-53-1	вторичный тантал
.	Некоторые корпуса компьютеров, дисковод и т.д.	алюминиевые сплавы
.	Корпуса стоек, ячеек, шкафов, компьютеров	сталь
.	Крепежные изделия	болты, гайки, винты
.	Вентиляторы и электромоторы	по паспорту СВТ
.	Пластиковая "фракция"	стеклотекстолит, пластмасса разъёмов и соединителей
0.	Экраны компьютеров	стеклофаза, содержащая Рв, Cd, CdS, редкоземельные металлы

Из данных табл.6 следует вывод о целесообразности извлечения вторичных чёрных и цветных металлов, пластмасс, стекла, крепёжных изделий, вентиляторов и электромоторов.

11.3.1.2.1. Извлечение вторичных чёрных металлов

Отечественная практика показывает, что на 1 г извлекаемого золота приходится около 1 кг лома чёрных металлов. В связи с высокой стоимостью транспортно-погрузочных работ рекомендуется производить отгрузку предприятиям-покупателям партий лома чёрных металлов весом не менее 10 тонн. Блоки, панели, съёмные кожухи, рамы, каркасы шкафов и стоек стационарных ЭВМ, изготовленные из стального нормализованного профиля или листа подвергаются сортировке, набираются в партии и реализуются.

Предпочтительно заключение договоров при условии, когда предприятие-покупатель своим транспортом вывозит вторичные металлы с территории предприятия-продавца.

3.4.3. Крепёжные изделия, заготовки стального профиля, листов, вентиляторы, электропускатели, кнопки, электрический кабель направляются на реализацию непосредственно в торговую сеть.

Опыт показывает, что денежные средства от реализации этих изделий не превышают 0,6 % от общей суммы.

11.3.1.2.2. Извлечение вторичных цветных металлов

11.3.1.2.2.1. Извлечение вторичных ломов, содержащих медь

В процессе разборки изделий СВТ образуется лом (содержащий медь) классификация которого должна проводиться по ГОСТ 1639.

В соответствии с ГОСТ 1639 медные шины целесообразно относить к классу А, группам I и II; латунь - к группам IV-VIII; бронзу - к группам XI-XII; отходы кабеля и проводов ПП следует относить к классу Г, группа XIII.

Все виды ломов необходимо сортировать по классам и группам, формировать в партии и реализовывать.

11.3.1.2.2.2. Извлечение вторичного алюминия и алюминиевых сплавов

В процессе разборки изделий СВТ алюминий и его сплавы обычно содержатся в типовых конструкциях изделий. По ГОСТ 1639 их следует относить к классам А3 и Б5.

Все виды отходов необходимо сортировать, формировать в партии и реализовывать.

11.3.1.2.2.3. Извлечение вторичных ломов, содержащих свинец

Свинцово-оловянные припои содержатся в печатных платах и их количество превышает количество золота в десятки раз.

Припои регенерируются при переработке печатных плат.

11.3.1.2.2.4. Извлечение вторичных ломов, содержащих тантал

При разборке СВТ танталовые конденсаторы необходимо складировать отдельно для последующей реализации.

11.3.1.2.2.5. Переработка изделий из пластмасс

Пластмассы следует сортировать по видам.

Переработке подлежат термопласты: поливинилхлорид, полиэтилен, полистирол и т.п.

11.3.1.2.2.6. Переработка изделий из стекла

Стёкла люминесцентных экранов электронно-лучевых трубок следует использовать в производстве керамики и в качестве сырья при производстве новых люминесцентных трубок.

11.3.2. Реализация партий

Основные направления деятельности на этапе "Реализация партий" представлены на Рис. 11.4.

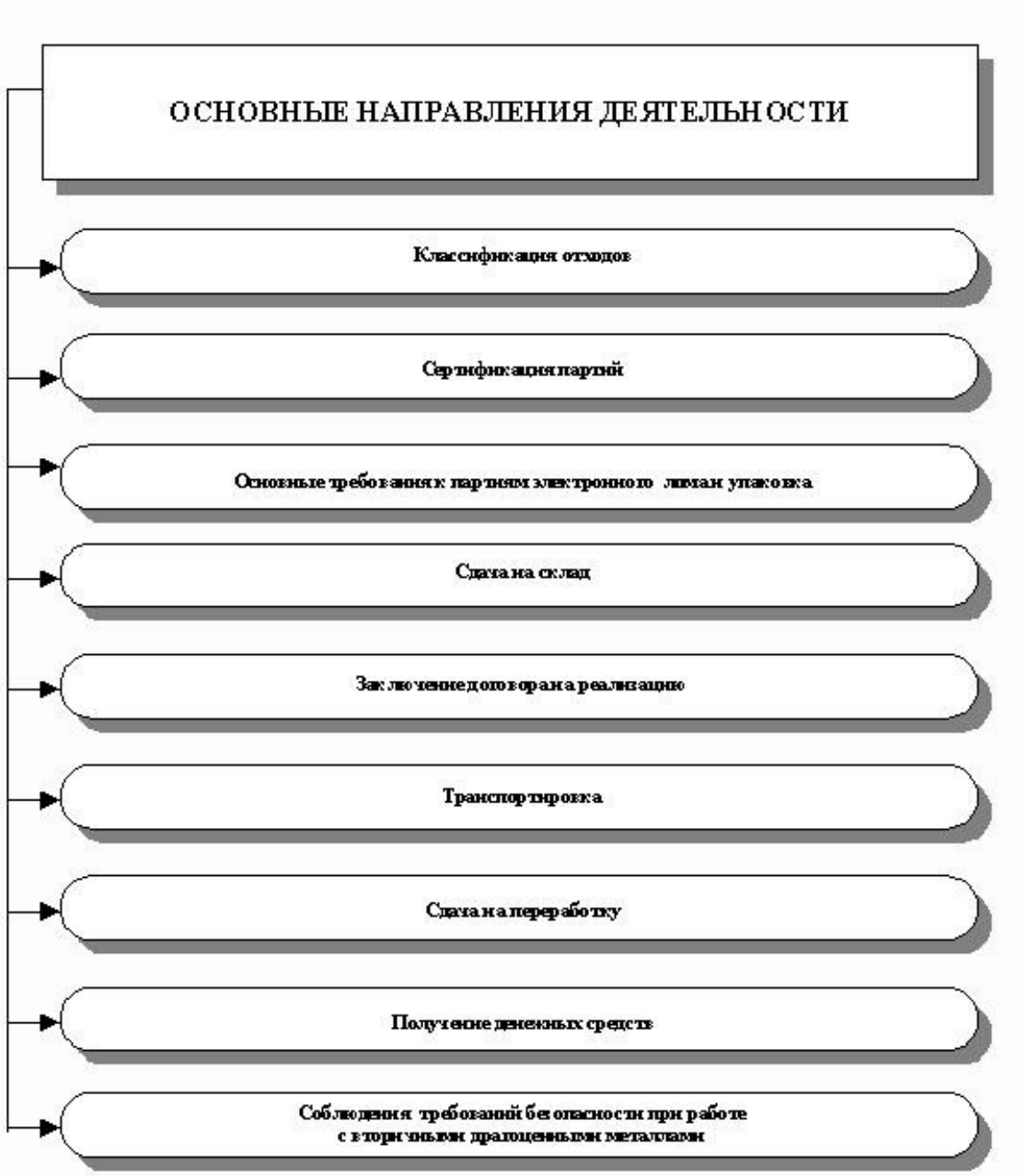

Рис. 11.4 Основные направления деятельности на этапе «Реализация партий»

Как видно из приведенной на рис.8 схемы, основные действия на этапе "Реализация партий" представляют собой последовательность действий, создающих основу для успешного выполнения процедур завершающего этапа утилизации СВТ.

11.3.2.1. Классификация отходов

В настоящее время в России и за рубежом не существует единой классификации вторичного сырья, содержащего драгоценные металлы. Поэтому, возможно разделение вторичного сырья по следующим признакам.

11.3.2.1.1. По содержанию драгоценных металлов:

- бедное (менее 1 % золота, 5 % серебра и 1 % металлов платиновой группы);
- богатое (более 1 % золота, 5 % серебра и 1 % металлов платиновой группы).
- По составу материала основания:
- на металлической основе;
- на органической (пластиковой) основе;
- на керамической основе;
- на комбинированной основе.
- По физическим признакам:
- твёрдые компактные отходы;
- сыпучие (порошки);
- жидкие.

11.3.2.1.2. По составу материала основания:

- на металлической основе;
- на органической (пластиковой) основе;
- на керамической основе;
- на комбинированной основе.

11.3.2.1.3. По физическим признакам:

- твёрдые компактные отходы;
- сыпучие (порошки);
- жидкие.

11.3.2.1.4. Возможна классификация вторичного сырья в зависимости от сферы производства в:

- ювелирной промышленности;
- химической промышленности;
- электронной, электрохимической, оборонной, радиопромышленности (радиолампы, разъёмы, контакты, контактные устройства, платы на органической основе, микросхемы, радиодетали, кабели и провода, лента, высечка, вырубка, аккумуляторы, элементы питания, прочие отходы);

- бытовых отходах (лом бытовой радиоэлектронной аппаратуры, бытовой стеклянный и фарфоровый бой, лом ювелирных украшений и т.д.).

Отходы классифицируются по элементному составу. При этом электронный лом отличается особым многообразием состава. Например, современный компьютерный лом содержит несколько десятков видов деталей, содержащих благородные, цветные и чёрные металлы.

Согласно методике опробования электронного лома и отходов, содержащих драгоценные металлы, разработанной ИАСЦ ГНЦ "ГИРЕДМЕТ", сырьё вторичных драгоценных металлов следует рассортировывать на классы, приведённые в табл.7.

Таблица 7

Классификация электронного лома и отходов, содержащих драгоценные металлы (по видам)

Номер класса	**Вид сырья**
1.	Микросхемы, транзисторы, диоды россыпью
2.	Конденсаторы россыпью
3.	Ножки разъёмов позолоченные и посеребренные
4.	Контакты разделанные
5.	Платы, содержащие элементы классов 1 и 2
6.	Разъёмы с позолоченными и посеребренными ножками
7.	Реле, переключатели, тумблеры
8.	Радиолампы
9.	Платы, содержащие элементы классов 1, 2, 6, 7 и 8
10.	Крупногабаритные детали (волноводы и т.п.)с покрытием из
11.	драгметаллами
12.	Элементы питания, аккумуляторы, ампульные батареи
	Сыпучие материалы (шихта катализаторов, зола фотоматериалов, шлам фиксажный и т.п.)

11.3.2.2. Сертификация партий

- В целях обеспечения строгого учёта, сохранности, сокращения потерь и эффективности использования драгоценных металлов, содержащихся в электронном ломе и отходах, а также для обеспечения единства и требуемой точности измерений при опробовании и проведении анализов химического состава, необходимо руководствоваться нормативными документами утверждёнными Комитетом Российской Федерации по драгоценным металлам и драгоценным камням, Комитетом Российской Федерации по стандартизации, метрологии и сертификации: "Порядком выдачи сертификатов химического состава на партии электронного лома и отходов, содержащих драгоценные металлы", "Временной методикой опробования электронного лома и отходов, содержащих драгоценные металлы".
- Указанные документы определяют порядок проведения работ и требования по опробованию и сертификации химического состава партий электронного лома и отходов.
- Сертификация химического состава электронного лома и отходов, содержащих драгоценные металлы, включает следующие работы:
    - оформление и представление заявки в соответствующий орган по сертификации;
    - создание комиссии по опробованию;
    - опробование, оформление документов по результатам опробирования, передача пробы на анализ;
    - собственно анализ пробы и оформление количественного химического анализа;
    - оформление сертификата.
- Выполнение измерений химического состава проб (анализ) электронного лома и отходов, содержащих драгоценные металлы, следует производить методами количественного химического анализа по аттестованным методикам, Приложение 7.
- Анализ проб осуществляется аналитическими лабораториями, которые аккредитованы Комитетом Российской Федерации по стандартизации, метрологии и сертификации, а также рекомендованными организациями и органами по сертификации, Приложение 8.
- По результатам анализа аккредитованная лаборатория оформляет протокол измерений химического состава пробы электронного лома или отходов по установленной форме.
- По результатам процедур опробования и анализа химического состава электронного лома или отходов, орган по сертификации оформляет и выдает заявителю Сертификат химического состава на содержание драгоценных металлов установленной формы.

- Сертификат химического состава и комплекс документов по опробованию сертифицируемой партии электронного лома включается в состав сопроводительной документации при передаче партии вторичного сырья от сдатчика заготовителю или переработчику, а также при вывозе за границу для переработки.
- При возникновении разногласий, в процессе передачи сертифицированных партий электронного лома и отходов от сдатчика заготовителю или переработчику, производится арбитраж. В этом случае партия не может быть передана переработчику до получения заключения арбитражной лаборатории, аккредитованной Комитетом Российской Федерации по стандартизации, метрологии и сертификации.

11.3.2.3. Основные требования к партиям электронного лома и упаковка

Утверждённые технические требования к партиям электронного лома, в частности его упаковке, до настоящего времени отсутствуют.

В связи с этим рекомендуется придерживаться следующих правил.

- Не допускается смешивание различных классов лома.
- В ломе и отходах драгоценных металлов не допускаются посторонние предметы, не относящиеся к естественной засорённости.
- Лом и отходы драгоценных металлов должны храниться в специально предназначенной для этого таре: в пакетах из полиэтиленовой плёнки по ГОСТ 10354, в фанерных ящиках по ГОСТ 9396, в металлических ящиках с замками собственного изготовления или мешках из мешочной бумаги по ГОСТ 2226.

Ящики внутри должны быть выложены упаковочной бумагой по ГОСТ 515 или каким-либо плёночным материалом. Пакеты и мешки, предназначенные для хранения и отправки лома и отходов, должны изготавливаться с вывернутыми внутрь двумя боковыми швами без нижнего шва.

Мешки или пакеты, уложенные в деревянные или металлические ящики, допускается заклеивать какой-либо клеевой лентой по ГОСТ 18351.

- Опечатывание отходов в таре производится после прошивания мешков и пакетов шпагатом по ГОСТ 17308 или заваривания открытых краев полиэтиленовых пакетов с последующим складыванием краёв "гармошкой" и прошиванием её двумя концами шпагата.

- Партия лома и отходов должна состоять из одного или нескольких мест в каждом из которых вложена одна или несколько позиций различающихся по составу компонентов, конфигураций, габаритным размерам и другими признаками, не меняющих принципиально сущности последующего опробования на перерабатывающих заводах.
- Взвешивание и упаковка отходов производится с участием материально ответственных лиц подразделений предприятия сдатчика.

После контрольного взвешивания каждое место должно быть опломбировано или опечатано сургучной печатью сдатчика.

В сопроводительных документах указывается описание пломбы или печати.

11.3.2.4. Сдача на склад

Передача партии электронного лома из производственного подразделения на склад драгоценных металлов предприятия осуществляется на основе приёмно-передаточного акта, акта изъятия узлов и изделий техники и других нормативных документов.

11.3.2.5. Заключение договора на реализацию

Между продавцом и покупателем заключается типовой договор, предметом которого является полученная партия электронного лома.

В договоре указываются обязанности сторон, порядок подготовки, отправки лома и отходов драгоценных металлов, порядок приёмки, опробования лома и отходов, порядок расчётов и ответственность сторон.

11.3.2.6. Транспортировка

При транспортировке партий лома необходимо соблюдать следующие условия.

- Транспортирование лома и отходов драгоценных металлов с содержанием золота и металлов платиновой группы более 5 % должно производиться через специальную связь в соответствии с инструкцией Министерства связи Российской Федерации о перевозке ценностей.
- Лом и отходы с содержанием золота и металлов платиновой группы менее 5 %, а также отходы серебра отправляются на перерабатывающие заводы почтовыми посылками, багажом по железной дороге или другим видом транспорта с оценочной стоимостью отгружаемого груза.
- Сдача на переработку.
- Порядок сдачи партии на переработку определяется условиями договора. Типичные условия договора для завода ВДМ следующие.

- Вскрытие посылок (мест) производится на заводе приёмной комиссией, которая взвешивает и сверяет фактическое наличие сырья и его качественный состав по каждому виду сырья с данными продавца. По результатам приёмки сырья покупатель высылает продавцу приёмный акт в течение 15 дней от даты поступления сырья.
- При доставке сырья транспортом продавца, приёмный акт на количество мест выдаётся на руки уполномоченному представителю продавца в день сдачи сырья.
- Представителю продавца необходимо иметь копию описи сдаваемого сырья.
- В случае нарушения упаковки или целостности печати материально ответственный работник покупателя в акте на приём отходов отмечает все нарушения.
- При расхождении фактически установленных данных при приёмке сырья с данными, значившимися в сопроводительных документах продавца, а также при отсутствии сопроводительных документов, окончательными результатами приёмки является масса брутто, нетто сырья, установленные приёмной комиссией покупателя.
- Взвешивание, опробование, пробоотбор и химический анализ проб каждой партии производится в соответствии с нормативно-технической документацией и по технологии, применяемой покупателем.
- Однотипные позиции партии подлежат объединению и опробыванию по единой технологической схеме.
- По результатам опробования сырья на содержание драгоценных металлов, которое производится в течение 60 дней со дня его поступления, покупатель высылает продавцу паспорт с указанием количества драгоценного металла с учетом процента выхода чистого металла в готовую продукцию.

Паспорт подписывается руководителем предприятия-покупателя, главным бухгалтером или их заместителями и скрепляется печатью покупателя.

11.3.2.7. Получение денежных средств

Порядок получения денежных средств зависит от условия договора. Типичные условия завода ВДМ следующие.

- Стоимость поставленного продавцом сырья определяется паспортом покупателя, составленным на основании прейскуранта завода, по мировым ценам на продукцию, получаемую из сырья на день, предшествующий выписке паспорта и пересчитанным в рубли по курсу, установленному Центральным Банком Российской Федерации на день выписки паспорта.

- Денежные средства перечисляются на расчётный счет продавца в течение трёх банковских дней со дня получения подтверждающего документа о поступлении денежных средств на расчётный счёт покупателя.
- Продавец производит оплату с каждой поставленной партии за опробование.
- Все платежи по договору должны производиться в валюте Российской Федерации в безналичной форме.

11.3.2.8. Соблюдение требований безопасности при работе с вторичными драгоценными металлами

Выполнение работ по разборке списанных СВТ предполагает соблюдение общих правил, изложенных в инструкциях по охране труда для слесаря механо-сборочных работ и лиц, работающих с ручным электроинструментом.

Специальные требования техники безопасности при работе с вторичными драгоценными металлами следующие.

- Не допускается сбор, заготовка и переработка радиоактивного лома и отходов драгоценных металлов.
- Степень воздействия на организм человека вредных веществ, выделяющихся в процессе заготовки и переработки лома и отходов драгоценных металлов, класс опасности и их предельно-допустимая концентрация (ПДК) в воздухе рабочей зоны установлены по ГОСТ 12.1.005 и ГОСТ 12.1.007, табл.8.

Таблица 8

Степень воздействия драгоценных металлов на организм человека

Наименование металла	**Характер действия на организм человека**	**Пути проникновения**	**Класс опасности**	**ПДК вредных веществ в воздухе рабочей зоны, мг/м^3**
Серебро и его соединения	Отходы могут оказывать раздражающее действие на слизистую оболочку носа и дыхательных путей.	Органы дыхания	11	0,5-1
Золото, платина и его соединения	При длительном контакте могут вызывать аллергические дерматиты и экземы.	Открытые участки кожи	-	-
Рутений		Органы дыхания	11	-
Родий	Оказывает раздражающее действие на слизистую оболочку носа и дыхательных путей. У рабочих, занятых очисткой родия, иногда развивается сверхчувствительность	-	-	-

- Контроль за содержанием вредных веществ в воздухе рабочей зоны проводится в соответствии с требованиями ГОСТ 12.1.005 и ГОСТ 12.1.007. Анализ проб воздуха проводится в соответствии с нормативно-технической документацией, утверждённой Минздравом Российской Федерации.

- Производственные помещения в местах образования вредных веществ и пыли должны быть оборудованы вентиляцией согласно ГОСТ 12.4.021 с обеспечением санитарно-гигиенических требований к воздуху рабочей зоны.
- Для снятия статического электричества пылеприёмники, воздуховоды вентиляционных установок должны иметь заземление, выполненное и обозначенное в соответствии с ГОСТ 12.2.007.0, ГОСТ 12.2.007.14 и ГОСТ 21130.
- Для предотвращения попадания пыли, твёрдых веществ на слизистую оболочку глаз необходимо пользоваться защитными очками типа ПО-2, ПО-3 согласно ГОСТ 12.4.013.
- При работе с пылящими отходами необходимо пользоваться фильтрующими респираторами РУ-60 и РУ-60му по ГОСТ 17269 и респираторами "Лепесток" по ГОСТ 12.4.028.
- При этом респираторы должны периодически подвергаться промывке.
- Средства индивидуальной защиты работающих с ломом и отходами драгоценных металлов и сплавов должны соответствовать типовым отраслевым нормам бесплатной выдачи рабочим и служащим металлургических производств. Спецодежда должна соответствовать ГОСТ 29057 и ГОСТ 29058.
- Помещения в местах выгрузки и загрузки лома и отходов, оказывающих вредное воздействие на организм человека, должны быть оборудованы местными отсосами согласно ГОСТ 12.4.021.
- Производственные помещения должны соответствовать требованиям "Санитарных норм проектирования промышленных предприятий СН 245-71".
- Метеорологические условия производственных помещений должны соответствовать санитарным нормам проектирования промышленных предприятий по ГОСТ 12.1.005.
- Требования безопасности при погрузочно-разгрузочных работах лома и отходов драгоценных металлов и сплавов должны соответствовать ГОСТ 12.3.009.
- Требования по обеспечению взрывобезопасности. Предприятия и организации, заготавливающие и перерабатывающие лом и отходы драгоценных металлов сплавов, должны проверять весь лом и отходы драгоценных металлов на взрывобезопасность. Из лома необходимо отобрать и удалить взрывоопасные предметы, материалы, в том числе электронно-вакуумные трубки дисплеев. Замкнутые сосуды, резервуары и другие полые предметы (баллоны, цилиндры, сосуды, электровакуумные изделия и т.д.) разгерметизируются и освобождаются от содержимого (газов или жидкостей). Разгерметизацию должны производить рабочие, прошедшие специальное обучение, которые перед началом работы инструктируются в установленном порядке о мерах предосторожности.

11.4. Приложения

11.4.1. Приложение 1

Перечень продукции и контролируемые гигиенические параметры.

Таблица 1

п/п	Вид продукции	Код ОКП	Контролируемые гигиенические параметры
	Машины вычислительные электронные цифровые, машины вычислительные электронные цифровые персональные (включая портативные ЭВМ)	40 1300, 40 1350, 40 1370	Уровни электромагнитных полей (ЭМП), акустического шума, концентрация вредных веществ в воздухе, визуальные показатели ВДТ, мягкое рентгеновское излучение *
	Устройства периферийные: принтеры, сканеры, модемы, сетевые устройства, блоки бесперебойного питания и т.д.	40 3000	Уровни ЭМП, акустического шума, концентрация вредных веществ в воздухе
	Устройства отображения информации (видеодисплейные терминалы)	40 3200	Уровни ЭМП, визуальные показатели, концентрация вредных веществ в воздухе, мягкое рентгеновское излучение *
	Автоматы игровые с использованием ПЭВМ	96 8575	Уровни ЭМП акустического шума, концентрация вредных веществ в воздухе, визуальные показатели ВДТ, мягкое рентгеновское излучение ·

· Контроль мягкого рентгеновского излучения осуществляется только для видеодисплейных терминалов с использованием электронно-лучевых трубок.

Допустимые значения уровней звукового давления в октавных полосах частот и уровня звука, создаваемого ПЭВМ.

Таблица 2

Уровни звукового давления в октавных полосах со среднегеометрическими частотами									Уровни звука в дБА
31,5 Гц	63 Гц	125 Гц	250 Гц	500 Гц	1000 Гц	2000 Гц	4000 Гц	8000 Гц	
86 дБ	71 дБ	61 дБ	54 дБ	49 дБ	45 дБ	42 дБ	40 дБ	38 дБ	50

Измерение уровня звука и уровней звукового давления проводится на расстоянии 50 см от поверхности оборудования и на высоте расположения источника(ков) звука.

Временные допустимые уровни ЭМП, создаваемых ПЭВМ.

Временные допустимые уровни ЭМП, создаваемые ПЭВМ.

Таблица 3

Наименование параметров		ВДУ ЭМП
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Электростатический потенциал экрана видеомонитора		500 В

Допустимые визуальные параметры устройств отображения информации.

Таблица 4

п/п	Параметры	Допустимые значения
	Яркость белого поля	Не менее 35 кд/кв. м
	Неравномерность яркости рабочего поля	Не более +/- 20%

	Контрастность (для монохромного режима)	Не менее 3:1
	Временная нестабильность изображения (непреднамеренное изменение во времени яркости изображения на экране дисплея)	Не должна фиксироваться
	Пространственная нестабильность изображения (непреднамеренные изменения положения фрагментов изображения на экране)	Не более 2 x 1E(-4L), где L - проектное расстояние наблюдения, мм

Для дисплеев на ЭЛТ частота обновления изображения должна быть не менее 75 Гц при всех режимах разрешения экрана, гарантируемых нормативной документацией на конкретный тип дисплея, и не менее 60 Гц для дисплеев на плоских дискретных экранах (жидкокристаллических, плазменных и т.п.).

11.4.2. Приложение 2

Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах.

Таблица 1

Наименование параметров		ВДУ
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Напряженность электростатического поля		15 кВ/м

Визуальные параметры ВДТ, контролируемые на рабочих местах.

Таблица 2

п/п	Параметры	Допустимые значения
	Яркость белого поля	Не менее 35 кд/кв. м
	Неравномерность яркости рабочего поля	Не более +/- 20%
	Контрастность (для монохромного режима)	Не менее 3:1
	Временная нестабильность изображения (мелькания)	Не должна фиксироваться
	Пространственная нестабильность изображения (дрожание)	Не более 2 х 1E(-4L), где L - проектное расстояние наблюдения, мм

Оптимальные параметры микроклимата во всех типах учебных и дошкольных помещений с использованием ПЭВМ.

Таблица 3

Температура, град. С	Относительная влажность, %	Абсолютная влажность, г/м3	Скорость дви-жения воздуха, м/с
19	62	10	< 0,1
20	58	10	< 0,1
21	55	10	< 0,1

11.4.3. Приложение 3. Методика инструментального контроля и гигиенической оценки уровней электромагнитных полей на рабочих местах

1. Общие положения.

1.1. Инструментальный контроль электромагнитной обстановки на рабочих местах пользователей ПЭВМ производится:

- при вводе ПЭВМ в эксплуатацию и организации новых и реорганизации рабочих мест;
- после проведения организационно-технических мероприятий, направленных на нормализацию электромагнитной обстановки;
- при аттестации рабочих мест по условиям труда;
- по заявкам предприятий и организаций.

1.2. Инструментальный контроль осуществляется органами ГСЭН и (или) испытательными лабораториями (центрами), аккредитованными в установленном порядке.

2. Требования к средствам измерений.

2.1. Инструментальный контроль уровней ЭМП должен осуществляться приборами с допускаемой основной относительной погрешностью измерений +/- 20%, включенными в Государственный реестр средств измерения и имеющими действующие свидетельства о прохождении Государственной поверки.

2.2. Следует отдавать предпочтение измерителям с изотропными антеннами-преобразователями.

3. Подготовка к проведению инструментального контроля.

3.1. Составить план (эскиз) размещения рабочих мест пользователей ПЭВМ в помещении.

3.2. Занести в протокол сведения об оборудовании рабочего места –наименования устройств ПЭВМ, фирм-производителей, моделей и заводские (серийные) номера.

3.3. Занести в протокол сведения о наличии санитарно-эпидемиологического заключения на ПЭВМ и приэкранные фильтры (при их наличии).

3.4. Установить на экране ВДТ типичное для данного вида работы изображение (текст, графики и др.).

3.5. При проведении измерений должна быть включена вся вычислительная техника, ВДТ и другое используемое для работы электрооборудование, размещенное в данном помещении.

3.6. Измерения параметров электростатического поля проводить не ранее чем через 20 минут после включения ПЭВМ.

4. Проведение измерений.

4.1. Измерение уровней переменных электрических и магнитных полей, статических электрических полей на рабочем месте, оборудованном ПЭВМ, производится на расстоянии 50 см от экрана на трех уровнях на высоте 0,5 м, 1,0 м и 1,5 м.

5. Гигиеническая оценка уровней ЭМП на рабочих местах.

5.1. Гигиеническая оценка результатов измерений должна осуществляться с учетом погрешности используемого средства метрологического контроля.

5.2. Если на обследуемом рабочем месте, оборудованном ПЭВМ, интенсивность электрического и/или магнитного поля в диапазоне 5 - 2000 Гц превышает значения, приведенные в таблице 5, следует проводить измерения фоновых уровней ЭМП промышленной частоты (при выключенном оборудовании). Фоновый уровень электрического поля частотой 50 Гц не должен превышать 500 В/м. Фоновые уровни индукции магнитного поля не должны превышать значений, вызывающих нарушения требований к визуальным параметрам ВДТ.

11.4.4. Приложение 4. Высота одноместного стола для работы с ПЭВМ

Рост учащихся или студентов в обуви, см	Высота над полом	
	поверхность стола	пространство для ног, не менее
116-130	520	400
131-145	580	520
146-160	640	580
161-175	700	640
выше 175	760	700

Примечание: ширина и глубина пространства для ног определяются конструкцией стола.

11.4.5. Приложение 5. Основные размеры стула

Параметры стула	Рост учащихся и студентов в обуви, см				
	16-130	131-145	146-160	161-175	<175
Высота сиденья над полом, мм	00	340	380	420	460
Ширина сиденья, не менее, мм	70	290	320	340	360
Глубина сиденья, мм	90	330	360	380	400
Высота нижнего края спинки над сиденьем, мм	30	150	160	170	190
Высота верхнего края над сиденьем, мм	80	310	330	360	400
Высота линии прогиба спинки, не менее, мм	70	190	200	210	220
Радиус изгиба переднего края сиденья, мм	20-50				
Угол наклона сиденья, гр.	0-4				
Угол наклона спинки, гр.	95-108				
Радиус спинки в плане, не менее, мм	300				

11.4.6. Приложение 6. Перечень нормативно-правовых документов, регламентирующих работу с драгоценными металлами

- Постановление Правительства Российской Федерации от 25 июня 1992 года №431 "О порядке сбора, приёмки и переработки лома и отходов драгоценных металлов и драгоценных камней".
- Инструкция Минфина России от 4 августа 1992 года №67 "О порядке получения, расходования, учёта и хранения драгоценных металлов и драгоценных камней на предприятиях, в учреждениях и организациях".
- Постановление Правительства Российской Федерации от 24 января 1994 года №35 "О порядке ввоза в Российскую Федерацию и вывоза из Российской Федерации товаров с содержанием драгоценных металлов, драгоценных камней, янтаря и изделий из него".
- Постановление Правительства Российской Федерации от 25 мая 1994 года №548 "О Федеральной целевой программе промышленной утилизации вооружения и военной техники на период до 2000 года".
- Постановление Правительства Российской Федерации от 30 июня 1994 года №756 "Об утверждении Положения о совершенствовании сделок с драгоценными металлами на территории Российской Федерации".
- Временная инструкция "О порядке привлечения предприятий и организаций различных форм собственности для сбора и первичной переработки лома и отходов драгоценных металлов, находящихся в распоряжении Минобороны России, а также порядке учёта и контроля за их деятельностью".
- Распоряжение Правительства Российской Федерации от 18 января 1995 года №69-р.
- Информационное письмо Роскомдрагмета и Государственной фельдъегерской службы Российской Федерации от 28 февраля 1995 года №718/195 "О порядке транспортировки драгоценных металлов и драгоценных камней".
- Приказ от 5 июля 1995 года №161 "Об утверждении Порядка оформления в Роскомдрагмете согласования на вывоз лома и отходов, содержащих драгоценные металлы, для переработки на зарубежных предприятиях".
- Постановление Госкомстата Российской Федерации от 21 июня 1995 года №67 "Об утверждении форм федерального государственного статистического наблюдения за движением драгоценных металлов и драгоценных камней".
- Информационное письмо Минфина России от 29 сентября 1995 года № 107 "О порядке переоценки драгоценных металлов и драгоценных камней".

- Приказ Роскомдрагмета и Госстандарта России №40/48 от 14 февраля 1996 года "О введении в действие "Порядка выдачи сертификатов химического состава на партии электронного лома и отходов, содержащих драгоценные (благородные) металлы" и "Временной методики опробования электронного лома и отходов, содержащих драгоценные (благородные) металлы".
- Информационное письмо Госкомстата России от 12 мая 1996 года №24-1-21/966 "Об утверждении Роскомдрагметом инструкций по заполнению форм № 1-ДМ № 5-ДМ".
- Инструкция Роскомдрагмета от 4 июля 1996 года №15-051-181/15 по заполнению формы федерального государственного статистического наблюдения за остатками, поступлением и расходом драгоценных металлов и изделий из них (форма № 2-ДМ).
- Инструкция Роскомдрагмета от 4 июля 1996 года №15-151-181/14 по заполнению формы федерального государственного статистического наблюдения за остатками, поступлением и сдачей в Госфонд драгоценных металлов в виде лома и отходов (Приложение к форме № 2-ДМ).
- Инструкция Роскомдрагмета от 4 июля 1996 года №15-051-181/17 по заполнению формы федерального государственного статистического наблюдения за остатками, поступлением и расходом драгоценных металлов и их солей, полученных для выполнения давальческих заказов и централизованных поставок (форма № 2-ДМ давальческого сырья).
- Инструкция Роскомдрагмета от 4 июля 1996 года №15-051-181/13 по заполнению формы федерального государственного статистического наблюдения за остатками, поступлением и расходом драгоценных металлов, содержащихся в составе приборов, оборудования и других изделий (форма №4-ДМ).
- Инструкция Роскомдрагмета от 4 июля 1996 года №15-051-181/16 по заполнению формы федерального государственного статистического наблюдения за поступлением драгоценных металлов в Госфонд в виде лома и отходов (форма № 5-ДМ).
- Федеральный Закон от 26 марта 1998 года №41-ФЗ "О драгоценных металлах и драгоценных камнях".
- О внесении изменений в Федеральный закон "О драгоценных металлах и драгоценных камнях" от 31 марта 1999 года №66-ФЗ Москва, Кремль.

О сертификации драгоценных металлов, драгоценных камней и продукции из них. Постановление Правительства Российской Федерации от 5 апреля 1999 года №372.

11.4.7. Приложение 7. Перечень методик, рекомендуемых для количественного химического анализа электронного лома и отходов, содержащих драгоценные (благородные) металлы

Методика гравиметрического определения золота во вторичном сырье драгоценных (благородных) металлов	% мас.: 10-80
Методика атомно-абсорционного (с электротермической автомизацией) определения палладия, платины и золота во вторичном сырье драгоценных (благородных) металлов	% мас.: 0,0003-0,1
Методика атомно-эмиссионного (с индукционной плазмой) определения палладия, платины и золота во вторичном сырье драгоценных (благородных) металлов	% мас.: Pd: 0,005-10; Pt: 0,01-10 Au: 0,001-10
Методика рентгеноспектрального определения золота и серебра во вторичном сырье драгоценных (благородных) металлов	% мас.: Au:0,05-4 Ag 0,8-7,5
Методика НГ-активационного определения золота во вторичном сырье драгоценных (благородных) металлов	% мас.: 0,01-90
Методика НГ-определения золота и серебра во вторичном сырье драгоценных (благородных) металлов	масса в пробе: Au>0,005 г; Ag>1,0 г
Методика инструментального нейтронно-активационного определения золота и иридия во вторичном сырье драгоценных (благородных) металлов	масса в пробе: Au: 0,1-200 мг; It: 0,5-200 мг
Методика титрометрического определения серебра во вторичном сырье драгоценных (благородных) металлов	% мас.: 5-90

11.4.8. Приложение 8. Рекомендуемый перечень организаций, проводящих анализ материалов на содержание драгоценных металлов и Органов по сертификации электронного лома и отходов, содержащих драгоценные (благородные) металлы

n.n.	Наименование организации
I. Организации, проводящие анализ материалов на содержание драгоценных металлов	
.1.	Ассоциация "Аналитика" (организации - члены Ассоциации, аккредитованные в данной области)
.2.	Государственный научный центр - государственный научно-исследовательский и проектный институт редкометаллической промышленности (Гиредмет)
.3.	Всероссийский научно-исследовательский институт минерального сырья (ВИМС)
.4.	Центральный научно-исследовательский и геологоразведочный институт цветных и благородных металлов (ЦНИИГРИ)
.5.	Государственный научный центр - Государственный научно-исследовательский институт цветных металлов (Гинцветмет)
.6.	Иркутский научно-исследовательский институт благородных и редких металлов и алмазов (Иргиредмет)
.7.	Государственный проектный и научно-исследовательский институт никелево-кобальтовой промышленности (Гипроникель)
.8.	Испытательный центр "Механобр-Аналит"
II. Органы по сертификации электронного лома и отходов, содержащих драгоценные (благородные) металлы	
.1.	Ассоциация "Аналитика" (организации - члены Ассоциации, аккредитованные в данной области)
.2.	Государственный научный центр - государственный научно-исследовательский и проектный институт редкометаллической промышленности (Гиредмет)
.3.	Всероссийский научно-исследовательский институт минерального сырья (ВИМС)

.4.	Иркутский научно-исследовательский институт благородных и редких металлов и алмазов (Иргиредмет)
.5.	Государственный проектный и научно-исследовательский институт никелево-кобальтовой промышленности (Гипроникель)
.6.	ТОО "АНСЕРТЭКО"
.7.	АОЗТ "РОСВТОРДРАГМЕТ"
.8.	АО "УРАЛРЕДМЕТ"
.9.	Испытательный центр "ДИАКОМ"

12. ЗАКЛЮЧЕНИЕ

В рамках данного дипломного проекта был разработан язык процедурного программирования в системе имитационного РДО. Данная разработка позволила упростить реализацию сложных алгоритмических зависимостей, существенно сократив исходный код модели. Результат разработки можно наблюдать на примере описания части функций в модели, описывающей работу почтового отделения, текст приведён в приложении.

13. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. **Б., Страуструп.** *Язык програмирования C++. Специальное издание. Пер. с англ.* М. : ООО "Бином-Пресс", 2008 г. 1104 с.: ил.

2. **М., Фаулер и К., Скотт.** *UML. Основы. - Пер. с англ.* СПб. : Символ-Плюс, 2002. 192 с., ил.

3. *Справка по языку РДО.* (в составе программы).

4. **Емельянов, В. В. и Ясиновский, С. И.** *Имитационное моделирование систем: Учеб. пособие.* М. : МГТУ им. Н.Э.Баумана, 2009. 584 с.: ил (Информатика в техническом университете).

5. *Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.201-78.*

6. **Арсеньев, В. В. и Сажин, Ю. Б.** *Методические указания к выполнению организационно-экономической части дипломных проектов по созданию программной продукции.* М. : МГТУ им. Н.Э.Баумана, 1994. 52 с.

7. *RAO-Studio - Руководство пользователя.* 2007 г.

14. СПИСОК ИСПОЛЬЗОВАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

- RAO-Studio.
- Microsoft ©, Visual Studio 2005.
- Visual Paradigm ©, Visual Paradigm for UML.
- iGrafx ©, iGrafx.
- АСКОН ©, Компас3D V11.
- Parametric Technology Corporation ©, MathCad 14.
- Microsoft ©, Microsoft office.

15. ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ КОД ТЕСТОВЫХ ПРОГРАММ

15.1. Тестовая программа первого варианта реализации

```
//! ====================================
#include <list>
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <windows.h>
//! ====================================

typedef unsigned int ruint;

//#define USE_LOG
const ruint times = 10000;

//! ====================================
class break_Exseption
{
public:
 break_Exseption()
 {}
};
class return_Exseption
{
public:
 return_Exseption()
 {}
};

//! ====================================
class ICalc
{
public:
 virtual void onCalc() = 0;
};
typedef ICalc*             LPICalc;
typedef std::list<LPICalc> CalcList;

//! ====================================
class CalcListCalc: public ICalc
{
public:
 CalcListCalc(CalcList& calcList)
       : m_calcList(calcList)
 {}

protected:
 virtual void onCalc()
 {
       CalcList::iterator it = m_calcList.begin();
       while (it != m_calcList.end())
       {
             try
             {
                   (*it)->onCalc();
             }
             catch(break_Exseption)
             {
                   return;
             }
```

```
            catch(return_Exseption)
            {
                  throw return_Exseption();
            }
            ++it;
      }
      return;
 }

 CalcList m_calcList;
};

//! ======================================
class PatternCalc: public CalcListCalc
{
public:
 PatternCalc(CalcList& calcList)
      : CalcListCalc(calcList)
 {}

protected:
 virtual void onCalc()
 {
      try
      {
            CalcListCalc::onCalc();
      }
      catch(return_Exseption)
      {
            return;
      }
      return;
 }
};

//! ======================================
class EmptyCalc: public ICalc
{
public:
 EmptyCalc(ruint id)
      : m_id(id)
 {}

private:
 virtual void onCalc()
 {
#ifdef USE_LOG
      std::cout << m_id << std::endl;
#endif
      return;
 }

 ruint m_id;
};

//! ======================================
class BreakCalc: public ICalc
{
private:
 virtual void onCalc()
 {
#ifdef USE_LOG
      std::cout << "break" << std::endl;
```

```
#endif
	throw break_Exseption();
 }
};

class ReturnCalc: public ICalc
{
private:
 virtual void onCalc()
 {
#ifdef USE_LOG
	std::cout << "return" << std::endl;
#endif
	throw return_Exseption();
 }
};

//! ======================================
ruint getMSec(const SYSTEMTIME& systime)
{
 return systime.wMilliseconds + systime.wSecond * 1000 + systime.wMinute *
1000 * 60 + systime.wHour * 1000 * 60 * 60;
}

//! ======================================
float sred(ruint a[10])
{
 ruint sred = 0;
 for(ruint i = 0; i < 10; i++)
 {
	sred += a[i];
 }
 return sred/10;
}
//! ======================================
void main()
{
	EmptyCalc  calc11(11) ;
	EmptyCalc  calc12(12) ;
	EmptyCalc  calc13(13) ;
	BreakCalc  breakCalc14;
	EmptyCalc  calc15(15) ;

 CalcList calcList1;
 calcList1.push_back(&calc11);
 calcList1.push_back(&calc12);
 calcList1.push_back(&calc13);
 calcList1.push_back(&breakCalc14);
 calcList1.push_back(&calc15);
 CalcListCalc calcListCalc1(calcList1);

 EmptyCalc  calc2(2);
 EmptyCalc  calc3(3);


	EmptyCalc  calc41(41)  ;
	EmptyCalc  calc42(42)  ;
	ReturnCalc returnCalc43;
	EmptyCalc  calc44(44)  ;
	EmptyCalc  calc45(45)  ;

 CalcList calcList4;
 calcList4.push_back(&calc41);
```

```
calcList4.push_back(&calc42);
calcList4.push_back(&returnCalc43);
calcList4.push_back(&calc44);
calcList4.push_back(&calc45);
CalcListCalc calcListCalc4(calcList4);

EmptyCalc  calc5(5);

CalcList calcList;
calcList.push_back(&calcListCalc1);
calcList.push_back(&calc2);
calcList.push_back(&calc3);
calcList.push_back(&calcListCalc4);
calcList.push_back(&calc5);

PatternCalc patternCalc(calcList);

LPICalc pCalc = &patternCalc;

char garbageArray[500][500];
for(ruint j = 0; j < 100; j++)
{

      for(ruint i = 0; i < 100; i++)
      {
            garbageArray[j][i] = (char)rand();
      }
}

int timeWork[10][30];

for(ruint k = 0; k < 10; k++)
{
      std::ofstream garbageFile("C:/test/garbageFile.txt",
std::ios_base::app);
      for(ruint j = 1; j <= 30; j++)
      {
            SYSTEMTIME timeBefore;
            ::GetSystemTime(&timeBefore);

            for (ruint i = 0; i < times*j; ++i)
            {
                  pCalc->onCalc();
            }

            SYSTEMTIME timeAfter;
            ::GetSystemTime(&timeAfter);

            ruint delay = getMSec(timeAfter) - getMSec(timeBefore);
            for(ruint t = 0; t < 100; t++)
            {

                  for(ruint r = 0; r < 100; r++)
                  {
                        garbageFile <<garbageArray[t][r]<< std::endl;
                  }
            }

            timeWork[k][j] = delay;
            std::cout <<j<<" ";
      }
      garbageFile.close();
      std::cout << "\a\a\a"<<k<< std::endl;
```

```
 }
 std::ofstream myFile("C:/test/exp_test.txt", std::ios_base::app);
 for(ruint q = 1; q <= 30; q++)
 {
       ruint asd[] = {timeWork[0][q], timeWork[1][q], timeWork[2][q],
timeWork[3][q], timeWork[4][q], timeWork[5][q], timeWork[6][q],
timeWork[7][q], timeWork[8][q], timeWork[9][q]};
       myFile << "delay "<<q<< " = " << sred(asd) << std::endl;
 }
 myFile.close();
}
```

15.2. Тестовая программа второго варианта реализации

```
//! ======================================
#include <list>
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <windows.h>
#include <string>
//! ======================================

typedef unsigned int ruint;

//#define USE_LOG
const ruint times = 10000;

//! ======================================
enum
{
 S_CONTINUE = 0,
 S_BREAK,
 S_RETURN
};

static unsigned int s_state = S_CONTINUE;

//! ======================================
class ICalc
{
public:
 virtual void onCalc() = 0;
};
typedef ICalc*             LPICalc;
typedef std::list<LPICalc> CalcList;

//! ======================================
class CalcListCalc: public ICalc
{
public:
 CalcListCalc(CalcList& calcList)
       : m_calcList(calcList)
 {}

protected:
 virtual void onCalc()
 {
       CalcList::iterator it = m_calcList.begin();
       while (it != m_calcList.end())
       {
             (*it)->onCalc();
             if (s_state != S_CONTINUE)
```

```
            {
                  break;
            }
            ++it;
      }

      if (s_state == S_BREAK)
      {
            s_state = S_CONTINUE;
      }
      return;
 }

 CalcList m_calcList;
};

//! ======================================
class PatternCalc: public CalcListCalc
{
public:
 PatternCalc(CalcList& calcList)
      : CalcListCalc(calcList)
 {}

protected:
 virtual void onCalc()
 {
      CalcListCalc::onCalc();
      if (s_state == S_RETURN)
      {
            s_state = S_CONTINUE;
      }
      return;
 }
};

//! ======================================
class EmptyCalc: public ICalc
{
public:
 EmptyCalc(ruint id)
      : m_id(id)
 {}

private:
 virtual void onCalc()
 {
#ifdef USE_LOG
      std::cout << m_id << std::endl;
#endif
      return;
 }

 ruint m_id;
};

//! ======================================
class BreakCalc: public ICalc
{
private:
 virtual void onCalc()
 {
      s_state = S_BREAK;
```

```
#ifdef USE_LOG
       std::cout << "break" << std::endl;
#endif
       return;
 }
};

class ReturnCalc: public ICalc
{
private:
 virtual void onCalc()
 {
       s_state = S_RETURN;
#ifdef USE_LOG
       std::cout << "return" << std::endl;
#endif
       return;
 }
};

//! ======================================
ruint getMSec(const SYSTEMTIME& systime)
{
 return systime.wMilliseconds + systime.wSecond * 1000 + systime.wMinute *
1000 * 60 + systime.wHour * 1000 * 60 * 60;
}

//! ======================================
float sred(ruint a[10])
{
 ruint sred = 0;
 for(ruint i = 0; i < 10; i++)
 {
       sred += a[i];
 }
 return sred/10;
}
//! ======================================
void main()
{
 EmptyCalc  calc11(11) ;
 EmptyCalc  calc12(12) ;
 EmptyCalc  calc13(13) ;
 BreakCalc  breakCalc14;
 EmptyCalc  calc15(15) ;

 CalcList calcList1;
 calcList1.push_back(&calc11);
 calcList1.push_back(&calc12);
 calcList1.push_back(&calc13);
 calcList1.push_back(&breakCalc14);
 calcList1.push_back(&calc15);
 CalcListCalc calcListCalc1(calcList1);

 EmptyCalc  calc2(2);
 EmptyCalc  calc3(3);


 EmptyCalc  calc41(41)  ;
 EmptyCalc  calc42(42)  ;
 ReturnCalc returnCalc43;
 EmptyCalc  calc44(44)  ;
 EmptyCalc  calc45(45)  ;
```

```
 CalcList calcList4;
 calcList4.push_back(&calc41);
 calcList4.push_back(&calc42);
 calcList4.push_back(&returnCalc43);
 calcList4.push_back(&calc44);
 calcList4.push_back(&calc45);
 CalcListCalc calcListCalc4(calcList4);

 EmptyCalc  calc5(5);

 CalcList calcList;
 calcList.push_back(&calcListCalc1);
 calcList.push_back(&calc2);
 calcList.push_back(&calc3);
 calcList.push_back(&calcListCalc4);
 calcList.push_back(&calc5);

 PatternCalc patternCalc(calcList);

 LPICalc pCalc = &patternCalc;

 char garbageArray[500][500];
 for(ruint j = 0; j < 100; j++)
 {

       for(ruint i = 0; i < 100; i++)
       {
             garbageArray[j][i] = (int)rand();
       }
 }

 int timeWork[10][30];

 for(ruint k = 0; k < 10; k++)
 {
       std::ofstream garbageFile("C:/test/garbageFile.txt",
std::ios_base::app);
       for(ruint j = 1; j <= 30; j++)
       {
             SYSTEMTIME timeBefore;
             ::GetSystemTime(&timeBefore);

             for (ruint i = 0; i < times*j; ++i)
             {
                   pCalc->onCalc();
             }

             SYSTEMTIME timeAfter;
             ::GetSystemTime(&timeAfter);

             ruint delay = getMSec(timeAfter) - getMSec(timeBefore);
             for(ruint t = 0; t < 100; t++)
             {

                   for(ruint r = 0; r < 100; r++)
                   {
                         garbageFile <<garbageArray[t][r]<< std::endl;
                   }
             }

             timeWork[k][j] = delay;
       }
```

```
        garbageFile.close();
        std::cout << "\a\a\a"<<k<< std::endl;
 }
 std::ofstream myFile("C:/test/if_test.txt", std::ios_base::app);
 for(ruint q = 1; q <= 30; q++)
 {
        ruint asd[] = {timeWork[0][q], timeWork[1][q], timeWork[2][q],
timeWork[3][q], timeWork[4][q], timeWork[5][q], timeWork[6][q],
timeWork[7][q], timeWork[8][q], timeWork[9][q]};
        myFile << "delay "<<q<< " = " << sred(asd) << std::endl;
 }
 myFile.close();
}
```

16. ПРИЛОЖЕНИЕ 2. ОПИСАНИЕ ФУНКЦИЙ В МОДЕЛИ ПОЧТОВОГО ОФИСА

16.1. Текст модели до внедрения разработки

```
$Function  Показать_Спрятать : such_as Смотрители.Разрешить
$Type = algorithmic
$Parameters
   Сейчас : such_as Смотрители.Разрешить
   Номер  : such_as Окна.Номер
$Body
  if (Сейчас == Да)  return Нет;
  if (Сейчас == Нет and
                 Номер == 1 and
                [Смотритель2.Разрешить == Да or
                 Смотритель3.Разрешить == Да or
                 Смотритель4.Разрешить == Да or
                 Смотритель5.Разрешить == Да ])
                           return Нет;
  if (Сейчас == Нет and
                 Номер == 1 and
                [Смотритель2.Разрешить == Нет and
                 Смотритель3.Разрешить == Нет and
                 Смотритель4.Разрешить == Нет and
                 Смотритель5.Разрешить == Нет ])
                           return Да;
  if (Сейчас == Нет and
                 Номер == 2 and
                [Смотритель1.Разрешить == Да or
                 Смотритель3.Разрешить == Да or
                 Смотритель4.Разрешить == Да or
                 Смотритель5.Разрешить == Да ])
                           return Нет;
  if (Сейчас == Нет and
                 Номер == 2 and
                [Смотритель1.Разрешить == Нет and
                 Смотритель3.Разрешить == Нет and
                 Смотритель4.Разрешить == Нет and
                 Смотритель5.Разрешить == Нет ])
                           return Да;
  if (Сейчас == Нет and
                 Номер == 3 and
                [Смотритель1.Разрешить == Да or
                 Смотритель2.Разрешить == Да or
                 Смотритель4.Разрешить == Да or
                 Смотритель5.Разрешить == Да ])
                           return Нет;
  if (Сейчас == Нет and
                 Номер == 3 and
                [Смотритель1.Разрешить == Нет and
                 Смотритель2.Разрешить == Нет and
                 Смотритель4.Разрешить == Нет and
                 Смотритель5.Разрешить == Нет ])
                           return Да;
  if (Сейчас == Нет and
                 Номер == 4 and
                [Смотритель1.Разрешить == Да or
                 Смотритель2.Разрешить == Да or
                 Смотритель3.Разрешить == Да or
                 Смотритель5.Разрешить == Да ])
                           return Нет;
```

```
  if (Сейчас == Нет and
                Номер == 4 and
               [Смотритель1.Разрешить == Нет and
                Смотритель2.Разрешить == Нет and
                Смотритель3.Разрешить == Нет and
                Смотритель5.Разрешить == Нет ])
                           return Да;
  if (Сейчас == Нет and
                Номер == 5 and
               [Смотритель1.Разрешить == Да or
                Смотритель2.Разрешить == Да or
                Смотритель3.Разрешить == Да or
                Смотритель4.Разрешить == Да ])
                           return Нет;
  if (Сейчас == Нет and
                Номер == 5 and
               [Смотритель1.Разрешить == Нет and
                Смотритель2.Разрешить == Нет and
                Смотритель3.Разрешить == Нет and
                Смотритель4.Разрешить == Нет ])
                           return Да;
$End

$Function Нормальный_Закон_Abs : real
$Type = algorithmic
$Parameters
  Результат : real
$Body
  if (Результат > 0) return Результат;
  if (Результат < 0) return -1 * Результат;
  if (Результат == 0) return 1/60;
$End

$Function Время_Прихода_Клиента  : real
$Type = algorithmic
$Parameters
  Вид    : such_as Виды_Нагрузки.Вид
  Число1 : real
  Число2 : real
$Body
  if (Вид == Постоянный)
                       return Число1;

  if (Вид == Линейный_Увеличение and
              Время.Часы == 9.0)
                       return Число1;
  if (Вид == Линейный_Увеличение and
              Время.Часы == 10.0)
                       return Число1 + 1 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 11.0)
                       return Число1 + 2 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 12.0)
                       return Число1 + 3 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 13.0)
                       return Число1 + 4 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
```

```
              Время.Часы == 14.0)
                      return Число1 + 5 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 15.0)
                      return Число1 + 6 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 16.0)
                      return Число1 + 7 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 17.0)
                      return Число1 + 8 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 18.0)
                      return Число1 + 9 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 19.0)
                      return Число1 + 10 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Увеличение and
              Время.Часы == 20.0)
                      return Число1 + 11 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);

  if (Вид == Линейный_Уменьшение and
              Время.Часы == Время_Рабочего_Дня + 9.0)
                      return Число1;
  if (Вид == Линейный_Уменьшение and
              Время.Часы == 9.0)
                      return Число2;
  if (Вид == Линейный_Уменьшение and
              Время.Часы == 10.0)
                      return Число2 -  1 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
              Время.Часы == 11.0)
                      return Число2 - 2 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
              Время.Часы == 12.0)
                      return Число2 - 3 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
              Время.Часы == 13.0)
                      return Число2 - 4 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
              Время.Часы == 14.0)
                      return Число2 - 5 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
              Время.Часы == 15.0)
                      return Число2 - 6 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
              Время.Часы == 16.0)
                      return Число2 - 7 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
```

```
                Время.Часы == 17.0)
                        return Число2 - 8 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
                Время.Часы == 18.0)
                        return Число2 - 9 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
                Время.Часы == 19.0)
                        return Число2 - 10 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Линейный_Уменьшение and
                Время.Часы == 20.0)
                        return Число2 - 11 * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  if (Вид == Равномерный)
                        return Равномерный_Закон(Число1,Число2);
  if (Вид == Нормальный)
                        return
Нормальный_Закон_Abs(Нормальный_Закон(Число1,Число2));
  if (Вид == Экспоненциальный)
                        return Экспоненциальный_Закон(Число1);
$End
```

16.2. Текст модели после внедрения разработки

```
$Function  Показать_Спрятать : such_as Смотрители.Разрешить
$Type = algorithmic
$Parameters
   Сейчас : such_as Смотрители.Разрешить
   Номер  : such_as Окна.Номер
$Body
  if (Сейчас == Да)  return Нет;
  for (integer i = 1; i <= 5; i++)
  {
   if (Сейчас == Нет and
                          Номер == i and
                         [Смотритель2.Разрешить == Да or
                          Смотритель3.Разрешить == Да or
                          Смотритель4.Разрешить == Да or
                          Смотритель5.Разрешить == Да ])
                                        return Нет;
   if (Сейчас == Нет and
                          Номер == i and
                         [Смотритель2.Разрешить == Нет and
                          Смотритель3.Разрешить == Нет and
                          Смотритель4.Разрешить == Нет and
                          Смотритель5.Разрешить == Нет ])
                                        return Да;
  }
$End

$Function Нормальный_Закон_Abs : real
$Type = algorithmic
$Parameters
  Результат : real
$Body
  if (Результат > 0) return Результат;
  if (Результат < 0) return -1 * Результат;
  if (Результат == 0) return 1/60;
$End

$Function Время_Прихода_Клиента  : real
```

```
$Type = algorithmic
$Parameters
  Вид    : such_as Виды_Нагрузки.Вид
  Число1 : real
  Число2 : real
$Body
  if (Вид == Постоянный)
                        return Число1;
  for(integer i = 0; i <= 11; i++)
  {
   if (Вид == Линейный_Увеличение and
                       Время.Часы == 9.0 + i)
                                 return Число1 + i * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  }

  if (Вид == Линейный_Уменьшение and
               Время.Часы == Время_Рабочего_Дня + 9.0)
                        return Число1;
  for(integer j = 0; j <= 11; j++)
  {
   if (Вид == Линейный_Уменьшение and
                       Время.Часы == 9.0 + j)
                                 return Число2 -  j * (Число2 - Число1) /
(Время_Рабочего_Дня - 1);
  }
  if (Вид == Равномерный)
                        return Равномерный_Закон(Число1,Число2);
  if (Вид == Нормальный)
                        return
Нормальный_Закон_Abs(Нормальный_Закон(Число1,Число2));
  if (Вид == Экспоненциальный)
                        return Экспоненциальный_Закон(Число1);
$End
```