Государственное образовательное учреждение высшего профессионального образования

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_\_РК (Робототехника и комплексная автоматизация)\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_РК9 (Компьютерные системы автоматизации производства)\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

\_\_\_Добавление наследования типов ресурсов в язык имитационного моделирования РДО\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_группа РК9-93\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** И.С. Намчук **\_\_**

(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта \_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**А.В. Урусов **\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Москва, 2010

Государственное образовательное учреждение высшего профессионального образования

***«Московский государственный технический университет имени Н.Э. Баумана»   
(МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_ г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине\_\_\_\_\_\_\_\_\_\_\_\_ Добавление наследования типов ресурсов в язык имитационного моделирования РДО \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Тема курсового проекта)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент\_\_\_\_\_\_\_\_\_Намчук И.С. РК9-93\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, инициалы, индекс группы)

График выполнения проекта: 25% к \_\_\_ нед., 50% к \_\_\_ нед., 75% к \_\_ нед., 100% к \_\_\_ нед.

1. Техническое задание

Добавить наследование типов ресурсов в язык имитационного моделирования РДО\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***2. Оформление курсового проекта***

2.1. Расчетно-пояснительная записка на 44 листах формата А4.

2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.)\_Лист 1: А1 Постановка задачи; Лист 2: А2 IDEF0 Компиляция RAO-Studio, А2 IDEF0 Декомпозиция Компиляция RAO-Studio; Лист 3: А2 IDEF0 Декомпозиция Компиляция rdo\_parser, А2 Диаграмма компонентов; Лист 4: А2 Блок-схема работы алгоритма, А2Синтаксическая диаграмма описания типов ресурсов; Лист 5: Результаты курсового проектирования\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_ 2009г.

**Руководитель курсового проекта** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_А.В. Урусов \_\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_** И.С. Намчук**\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Оглавление

[Введение 4](#_Toc253512777)

[1.Предпроектное исследование. 6](#_Toc253512778)

[1.1. Основные подходы к построению ИМ. 6](#_Toc253512779)

[1.2. Процесс имитации в РДО. 7](#_Toc253512780)

[1.3. Основные положения языка РДО. 9](#_Toc253512781)

[1.4. Постановка задачи. 11](#_Toc253512782)

[2. Концептуальный этап проектирования. 13](#_Toc253512783)

[2.1.Диаграмма компонентов. 13](#_Toc253512784)

[2.2.Структура логического вывода РДО. 14](#_Toc253512785)

[2.3.Техническое задание. 15](#_Toc253512786)

[2.3.1.Общие сведения. 15](#_Toc253512787)

[2.3.2.Назначение и цели развития системы. 16](#_Toc253512788)

[2.3.3.Характеристики объекта автоматизации. 16](#_Toc253512789)

[2.3.4.Требования к системе. 16](#_Toc253512790)

[3.1.Разработка синтаксиса описания типа ресурса. 18](#_Toc253512791)

[3.2.Разработка архитектуры компонента rdo\_parser. 19](#_Toc253512792)

[4.Рабочий этап проектирования. 20](#_Toc253512793)

[4.1.Синтаксический анализ типов данных. 20](#_Toc253512794)

[Заключение 21](#_Toc253512795)

[Список использованных источников 22](#_Toc253512796)

[Приложение 1. Модель гибкой производственной системы на языке РДО 23](#_Toc253512797)

[Приложение 2. Полный синтаксический анализ описания типа данных (rdortp.y). 33](#_Toc253512798)

# Введение

“Сложные системы”, “системность”, “бизнес-процессы”, “управление сложными системами”, “модели” – все эти термины в настоящее время широко используются практически во всех сферах деятельности человека. Причиной этого является обобщение накопленного опыта и результатов в различных сферах человеческой деятельности и естественное желание найти и использовать некоторые общесистемные принципы и методы. Именно системность решаемых задач в перспективе должна стать той базой, которая позволит исследователю работать с любой сложной системой, независимо от ее физической сущности. Именно модели и моделирование систем является тем инструментом, которое обеспечивает эту возможность.

Имитационное моделирование (ИМ) на ЭВМ находит широкое применение при исследовании и управлении сложными дискретными системами (СДС) и процессами в них. К таким системам можно отнести экономические и производственные объекты, морские порты, аэропорты, комплексы перекачки нефти и газа, программное обеспечение сложных систем управления, вычислительные сети и многие другие. Широкое использование ИМ объясняется сложностью (а иногда и невозможностью) применения строгих методов оптимизации, которая обусловлена размерностью решаемых задач и неформализуемостью сложных систем. Так выделяют, например, следующие проблемы в исследовании операций, которые не могут быть решены сейчас и в обозримом будущем без ИМ:

1. Формирование инвестиционной политики при перспективном планировании.
2. Выбор средств обслуживания (или оборудования) при текущем планировании.
3. Разработка планов с обратной информационной связью и операционных предписаний.

Эти классы задач определяются тем, что при их решении необходимо одновременно учитывать факторы неопределенности, динамическую взаимную обусловленность текущих решений и последующих событий, комплексную взаимозависимость между управляемыми переменными исследуемой системы, а часто и строго дискретную и четко определенную последовательность интервалов времени. Указанные особенности свойственны всем сложным системам.

Проведение имитационного эксперимента позволяет:

1. Сделать выводы о поведении СДС и ее особенностях:
   * без ее построения, если это проектируемая система;
   * без вмешательства в ее функционирование, если это действующая система, проведение экспериментов над которой или слишком дорого, или небезопасно;
   * без ее разрушения, если цель эксперимента состоит в определении пределов воздействия на систему.
2. Синтезировать и исследовать стратегии управления.
3. Прогнозировать и планировать функционирование системы в будущем.
4. Обучать и тренировать управленческий персонал и т.д.

ИМ является эффективным, но и не лишенным недостатков, методом. Трудности использования ИМ, связаны с обеспечением адекватности описания системы, интерпретацией результатов, обеспечением стохастической сходимости процесса моделирования, решением проблемы размерности и т.п. К проблемам применения ИМ следует отнести также и большую трудоемкость данного метода.

Интеллектуальное ИМ, характеризующееся возможностью использования методов искусственного интеллекта и, прежде всего, знаний, при принятии решений в процессе имитации, при управлении имитационным экспериментом, при реализации интерфейса пользователя, создании информационных банков ИМ, снимает часть проблем использования ИМ.

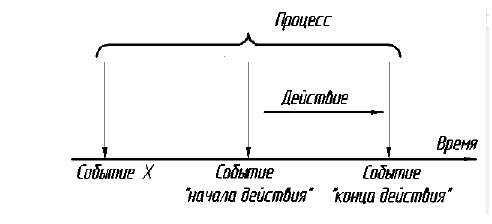
# 1.Предпроектное исследование.

## 1.1. 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. 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. 1.).



*Рис****. 1.*** *Взаимосвязь между событиями****,*** *действием и процессом****.***

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

## 1.2. Процесс имитации в РДО.

Для имитации работы модели в РДО реализованы два подхода: событийный и сканирования активностей.

***Событийный подход.***

При событийном подходе исследователь описывает события, которые могут изменять состояние системы, и определяет логические взаимосвязи между ними.

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.

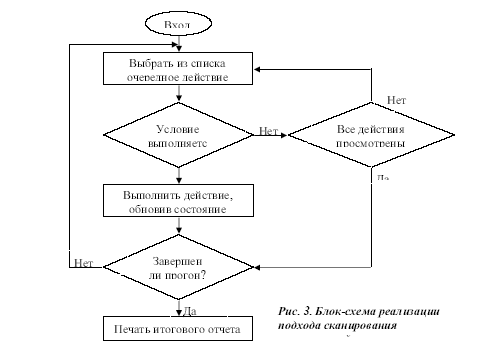
Например, событие начала обработки детали на станке приводит к появлению в списке будущих событий события окончания обработки детали, которое должно наступить в момент времени равный текущему времени плюс время, требуемое на обработку детали на станке. В событийных системах модельное время фиксируется только в моменты изменения состояний.



*Рис. 2. Выполнение событий в ИМ.*

***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. 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.



## 1.3. Основные положения языка РДО.

В основе системы РДО – «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).

## 1.4. Постановка задачи.

Основная идея курсового проект – добавления в язык имитационного моделирования РДО наследования типов ресурсов.

На данный момент в RAO Studio отсутствует наследование типов ресурсов, это лишает язык РДО необходимой гибкости.

Рассмотрим данный недостаток на примере модели гибкой производственной системы. ГПС состоит из трех станков, обслуживаемых тремя роботом, двух тележек и двух накопителей. Детали находятся в накопителе 1, их начальное количество равно 10. При помощи робота 1, детали помещаются на тележку 1. Тележка может транспортировать одновременно только одну деталь. Тележка транспортирует деталь к первой группе станков (Станок 1 и Станок 2), которые производят одну и туже операцию. Установка и съем детали производиться при помощи робота 2. По окончании обработки, деталь помещается на тележку 2, которая транспортирует деталь к станку 3. Установка на станок 3 производиться так же с помощью робота3. По окончании обработки деталь сбрасывается в накопитель 2

Модель данной гибкой производственной системы на языке РДО представлена в Приложении 1.

Рассмотрим описание типов ресурсов на вкладке RTP.

$Resource\_type Накопители : permanent

$Parameters

положение : (станок\_1, станок\_2, станок\_3, накопитель\_1, накопитель\_2, тележка\_1\_н, тележка\_1\_к, тележка\_2\_н, тележка\_2\_к, нигде)

номер : integer

максимальное\_количество : integer = 25

текущее\_количество : integer = 0

$End

$Resource\_type Тележки : permanent

$Parameters

номер : integer

положение : such\_as Накопители.положение

состояние : (свободен, занят, загружен, перемещается, прибыл, ожидает) = свободен

$End

$Resource\_type Роботы : permanent

$Parameters

номер : integer

положение : such\_as Накопители.положение

состояние : (свободен, занят) = свободен

$End

$Resource\_type Станки : permanent

$Parameters

номер : integer

положение : such\_as Накопители.положение

состояние : (свободен, загружается, готов\_к\_обработке, работает,разгружается, закончил\_обработку) = свободен

время\_работы : real

$End

$Resource\_type Детали : permanent

$Parameters

номер : integer

положение : such\_as Накопители.положение = накопитель\_1

состояние : (хранится, транспортируется, обрабатывается, обработка\_закончна) = хранится

$End

Видно, что в приведенной модели у всех типов ресурсов есть такие параметры как “номер” и “положение”, и для каждого типа ресурса они описываются отдельно. Это уменьшает гибкость модели и увеличивает трудоемкость написания модели.

Решением данной проблемы является использование наследования, с его помощью параметры родителя будут автоматически передаваться потомку.

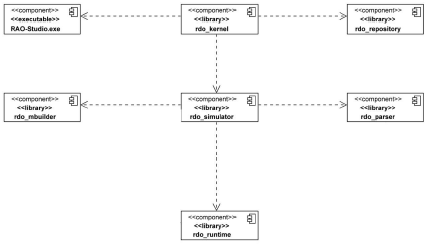
# 2. Концептуальный этап проектирования.

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.

## 2.1.Диаграмма компонентов.

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



*Рис. 4. Упрощенная диаграмма компонентов.*

Базовый функционал представленных на диаграмме компонентов:

***rdo\_kernel*** реализует ядровые функции системы. Не изменяется при разработке системы.

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

## 2.2.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.

## 2.3.Техническое задание.

### 2.3.1.Общие сведения.

В системе РДО разрабатывается наследование типов ресурсов. Основной разработчик РДО – кафедра РК-9, МГТУ им. Н.Э. Баумана.

### 2.3.2.Назначение и цели развития системы.

Основная цель данного курсового проекта – реализовать синтаксис описания наследования типов ресурсов в языке РДО, а так же организовать передачу параметров типов ресурсов от родителя потомкам.

### 2.3.3.Характеристики объекта автоматизации.

РДО – язык имитационного моделирования, включающий все три основные подхода описания дискретных систем: процессный, событийный и сканирования активностей.

### 2.3.4.Требования к системе.

При описании модели гибкой производственной системы в языке РДО, описание параметров “номер” и ”положение” производится однажды для новой сущности “Элементы\_участка”. Это обеспечивает более адекватное описание модели системы и сокращение кода.

Таким образом с учетом возможности наследования типов ресурсов вкладка RTP примет вид:

$Resource\_type Система : permanent

$Parameters

положение : (станок\_1, станок\_2, станок\_3, накопитель\_1, накопитель\_2, тележка\_1\_н, тележка\_1\_к, тележка\_2\_н, тележка\_2\_к, нигде)

$End

$Resource\_type Элементы\_участка : permanent

$Parameters

номер : integer = 1

положение : such\_as Система.положение

$End

$Resource\_type Накопители : Элементы\_участка : permanent

$Parameters

максимальное\_количество : integer = 25

текущее\_количество : integer = 0

$End

$Resource\_type Тележки : Элементы\_участка : permanent

$Parameters

состояние : (свободен, занят, загружен, перемещается, прибыл, ожидает) = свободен

$End

$Resource\_type Роботы : Элементы\_участка : permanent

$Parameters

состояние : (свободен, занят) = свободен

$End

$Resource\_type Станки : Элементы\_участка : permanent

$Parameters

состояние :(свободен, загружается, готов\_к\_обработке, работает,разгружается, закончил\_обработку) = свободен

время\_работы : real

$End

$Resource\_type Детали : permanent

$Parameters

номер : integer

положение : such\_as Система.положение = накопитель\_1

состояние :(хранится, транспортируется, обрабатывается, обработка\_закончна) = хранится

$End

## 3.1.Разработка синтаксиса описания типа ресурса.

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

Описание каждого типа ресурса имеет один из следующих форматов:

*$Resourse\_type<имя\_типа>:<вид\_ресурсов>*

*$Parameters*

*<описание\_параметра>{< описание\_параметра >}*

*$End*

либо

*$Resourse\_type<имя\_типа>:<имя\_типа\_родителя>:<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.

*имя\_типа\_родителя*

Имя типа родителя представляет собой простое имя. Имя типа родителя должно быть определено ранее.

*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 предопределеннымии 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;

ссылка на один из выше определенных типов - ***such\_as***.

## 3.2.Разработка архитектуры компонента rdo\_parser.

Для возможности обработки новой конструкции в коде модели требуют изменений лексический и синтаксический анализаторы РДО.

# 4.Рабочий этап проектирования.

## 4.1.Синтаксический анализ типов данных.

Для реализации в среде имитационного моделирования нового инструмента разработанного на концептуальном и техническом этапах проектирования необходимо добавить в генератор синтаксического анализатора (bison) описание наследования в rtp\_header:

RDO\_Resource\_type RDO\_IDENTIF\_COLON RDO\_IDENTIF\_COLON rtp\_vid\_res

{

LEXER->m\_enum\_param\_cnt = 0;

RDOValue\* type\_name = reinterpret\_cast<RDOValue\*>($2);

std::string name = type\_name->value().getIdentificator();

const RDORTPResType\* \_rtp = PARSER->findRTPResType( name );

if ( \_rtp ) {

PARSER->error\_push\_only( type\_name->src\_info(), rdo::format("Тип ресурса уже существует: %s", name.c\_str()) );

PARSER->error\_push\_only( \_rtp->src\_info(), "См. первое определение" );

PARSER->error\_push\_done();

}

RDOValue\* prnt\_type\_name = reinterpret\_cast<RDOValue\*>($3);

std::string prnt\_name = prnt\_type\_name->value().getIdentificator();

const RDORTPResType\* \_rtp\_prnt = PARSER->findRTPResType( prnt\_name );

if ( \_rtp\_prnt ) {

RDORTPResType\* rtp = new RDORTPResType( PARSER, type\_name->src\_info(), $4 != 0 );

rsint t\_ind=0, col\_par=\_rtp\_prnt->getParams().size();

while (t\_ind<col\_par)

{

rtp->addParam(\_rtp\_prnt->getParams()[t\_ind]);

PARSER->warning( rdo::format("Параметр %s передан от родителя %s потомку %s", \_rtp\_prnt->getParams()[t\_ind]->src\_info().src\_text().c\_str(), prnt\_name.c\_str(),name.c\_str()) );

t\_ind=t\_ind+1;

}

$$ = (int)rtp;

PARSER->warning( rdo::format("Тип ресурса %s является потомком типа ресурса %s", name.c\_str(), prnt\_name.c\_str()) );

}

else {

PARSER->error\_push\_only( rdo::format("Родительский тип ресурса не существует: %s", prnt\_name.c\_str()) );

PARSER->error\_push\_done();

}

}

Из этого кода можно сделать вывод, что параметры типа ресурса – родителя будут скопированы потомку при его создании.

# Заключение

В рамках данного курсового проекта были получены следующие результаты:

1) Проведено предпроектное исследование системы имитационного моделирования РДО и сформулированы предпосылки создания в системе наследования типов данных.

2) На этапе концептуального проектирования системы с помощью диаграммы компонентов нотации UML укрупнено, показано внутреннее устройство РДО и выделены те компоненты, которые потребуют внесения изменений в ходе этой работы. Разработаны функциональные диаграммы (в нотации IDEF0) процесса компиляции RAO-Studio.

3) На этапе технического проектирования разработан новый синтаксис для описания наследования типов ресурсов, который представлен на синтаксической диаграмме. Разработана блок – схема создаваемого алгоритма.

4) На этапе рабочего проектирования написан программный код для реализации изменеий в ***rdo\_parser*** системы РДО. Проведены отладка и тестирование новой системы, в ходе которых исправлялись найденные ошибки.

5) Измененная модель ГПС при моделировании дает теже результаты, что и исходная, т.е. добавление наследования не изменило логику работы модели.

Поставленная цель курсового проекта достигнута.

# Список использованных источников

1. RAO-Studio – Руководство пользователя, 2007 [**http://rdo.rk9.bmstu.ru/forum/viewtopic.php?t=900**].
2. Справка по языку РДО (в составе программы) [**http://rdo.rk9.bmstu.ru/forum/viewforum.php?f=15**].
3. Емельянов В.В., Ясиновский С.И. Имитационное моделирование систем: Учеб. пособие. – М.: Изд-во МГТУ им.Н.Э. Баумана, 2009. – 584с.: ил. (Информатика в техническом университете).
4. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.201-78.
5. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. ГОСТ 19.701-90. Условные обозначения и правила выполнения.
6. Бьерн Страуструп. Язык моделирования C++. Специальное издание. Пер. с англ. – М.: ООО «Бином-пресс», 2007 г. – 1104 с.: ил.

# Приложение 1. Модель гибкой производственной системы на языке РДО

**Primer.pat (Образцы):**

$Pattern Погрузка\_детали : operation trace

$Relevant\_resources

накопитель\_ : Накопители Keep NoChange

деталь\_ : Детали Keep Keep

робот\_ : Роботы Keep Keep

тележка\_ : Тележки Keep Keep

$Time = Экспоненциальный( Время\_погрузки )

$Body

накопитель\_

Choice from накопитель\_.номер = 1 and накопитель\_.текущее\_количество > 0

first

Convert\_begin

текущее\_количество set накопитель\_.текущее\_количество - 1

деталь\_

Choice from деталь\_.положение = накопитель\_1

first

Convert\_begin

положение set нигде

состояние set транспортируется

Convert\_end

положение set тележка\_1\_н

робот\_

Choice from робот\_.номер = 1 and робот\_.положение = накопитель\_1 and робот\_.состояние = свободен

first

Convert\_begin

положение set нигде

состояние set занят

Convert\_end

положение set тележка\_1\_н

состояние set свободен

тележка\_

Choice from тележка\_.номер = 1 and тележка\_.положение = тележка\_1\_н and тележка\_.состояние = свободен

first

Convert\_begin

состояние set занят

Convert\_end

состояние set загружен

$End

$Pattern Доставка\_детали : operation trace

$Parameters

номер\_тележки : integer

начальное\_положение\_устройства : such\_as Накопители.положение

конечное\_положение\_устройства : such\_as Накопители.положение

$Relevant\_resources

тележка\_ : Тележки Keep Keep

деталь\_ : Детали NoChange Keep

$Time = Экспоненциальный( Время\_доставки )

$Body

тележка\_

Choice from тележка\_.положение = начальное\_положение\_устройства

and тележка\_.состояние = загружен

first

Convert\_begin

состояние set перемещается

Convert\_end

положение set конечное\_положение\_устройства

состояние set прибыл

деталь\_

Choice from деталь\_.положение = начальное\_положение\_устройства

first

Convert\_end

положение set конечное\_положение\_устройства

$End

$Pattern Возврат\_робота : operation trace

$Relevant\_resources

робот\_ : Роботы Keep Keep

$Time = Время\_возврата

$Body

робот\_

Choice from робот\_.положение <> место\_возврата(робот\_.положение)

and робот\_.состояние = свободен

first

Convert\_begin

состояние set занят

Convert\_end

состояние set свободен

положение set место\_возврата(робот\_.положение)

$End

$Pattern Возврат\_тележки : operation trace

$Parameters

номер\_тележки : integer

начальное\_положение\_устройства : such\_as Накопители.положение

конечное\_положение\_устройства : such\_as Накопители.положение

$Relevant\_resources

тележка\_ : Тележки Keep Keep

$Time = Экспоненциальный( Время\_доставки )

$Body

тележка\_

Choice from тележка\_.положение = начальное\_положение\_устройства

and тележка\_.состояние = свободен

first

Convert\_begin

состояние set занят

Convert\_end

состояние set свободен

положение set конечное\_положение\_устройства

$End

$Pattern Установка\_на\_станке : operation trace

$Parameters

номер\_станка : such\_as Накопители.положение

положение\_устройства : such\_as Накопители.положение

$Relevant\_resources

станок\_ : Станки Keep Keep

тележка\_ : Тележки Keep Keep

робот\_ : Роботы Keep Keep

деталь\_ : Детали NoChange Keep

$Time = Экспоненциальный( Время\_установки\_на\_станок )

$Body

станок\_

Choice from станок\_.положение = номер\_станка and станок\_.состояние = свободен

first

Convert\_begin

состояние set загружается

Convert\_end

состояние set готов\_к\_обработке

тележка\_

Choice from тележка\_.положение = положение\_устройства

and тележка\_.состояние = прибыл

first

Convert\_begin

состояние set ожидает

Convert\_end

состояние set свободен

робот\_

Choice from робот\_.положение = положение\_устройства

and робот\_.состояние = свободен

first

Convert\_begin

состояние set занят

Convert\_end

состояние set свободен

положение set станок\_.положение

деталь\_

Choice from деталь\_.положение = положение\_устройства

first

Convert\_end

положение set номер\_станка

$End

$Pattern Обработка\_на\_станке : operation trace

$Parameters

номер\_станка : such\_as Накопители.положение

$Relevant\_resources

станок\_ : Станки Keep Keep

деталь\_ : Детали Keep Keep

$Time = станок\_.время\_работы

$Body

станок\_

Choice from станок\_.положение = номер\_станка and станок\_.состояние = готов\_к\_обработке

first

Convert\_begin

состояние set работает

Convert\_end

состояние set закончил\_обработку

деталь\_

Choice from деталь\_.положение = номер\_станка

first

Convert\_begin

состояние set обрабатывается

Convert\_end

состояние set обработка\_закончна

$End

$Pattern Разгрузка\_станков : operation trace

$Parameters

номер\_станка : such\_as Накопители.положение

$Relevant\_resources

станок\_ : Станки Keep Keep

робот\_ : Роботы Keep Keep

тележка\_ : Тележки Keep Keep

деталь\_ : Детали Keep Keep

$Time = Экспоненциальный( Время\_разгрузки\_станка )

$Body

станок\_

Choice from станок\_.положение = номер\_станка and станок\_.состояние = закончил\_обработку

first

Convert\_begin

состояние set разгружается

Convert\_end

состояние set свободен

робот\_

Choice from робот\_.состояние = свободен and робот\_станок(робот\_.номер, с танок\_.номер) = 1

first

Convert\_begin

состояние set занят

Convert\_end

состояние set свободен

положение set тележка\_2\_н

тележка\_

Choice from тележка\_.номер = 2 and тележка\_.состояние = свободен

first

Convert\_begin

состояние set занят

Convert\_end

состояние set загружен

деталь\_

Choice from деталь\_.положение = номер\_станка

first

Convert\_begin

состояние set транспортируется

Convert\_end

положение set тележка\_2\_н

$End

$Pattern Окончание\_обработки : operation trace

$Parameters

номер\_станка : such\_as Накопители.положение

$Relevant\_resources

станок\_ : Станки Keep Keep

деталь\_ : Детали Keep Keep

накопитель\_ : Накопитель\_2 NoChange Keep

$Time = Экспоненциальный( Время\_разгрузки\_станка )

$Body

станок\_

Choice from станок\_.положение = номер\_станка and станок\_.состояние = закончил\_обработку

first

Convert\_begin

состояние set разгружается

Convert\_end

состояние set свободен

деталь\_

Choice from деталь\_.положение = номер\_станка

first

Convert\_begin

состояние set транспортируется

Convert\_end

положение set накопитель\_2

накопитель\_

Choice NoCheck

first

Convert\_end

текущее\_количество set накопитель\_.текущее\_количество + 1

$End

$Pattern Работа\_таймера : irregular\_event trace

$Relevant\_resources

накопитель : Накопитель\_1 Keep

$Time = 0.5

$Body

накопитель

Convert\_event

положение set накопитель.положение

$End

**Primer.rtp (Типы ресурсов):**

$Resource\_type Система : permanent

$Parameters

положение : (станок\_1, станок\_2, станок\_3, накопитель\_1, накопитель\_2, тележка\_1\_н, тележка\_1\_к, тележка\_2\_н, тележка\_2\_к, нигде)

$End

$Resource\_type Элементы\_участка : permanent

$Parameters

номер : integer = 1

положение : such\_as Система.положение

$End

$Resource\_type Накопители : Элементы\_участка : permanent

$Parameters

максимальное\_количество : integer = 25

текущее\_количество : integer = 0

$End

$Resource\_type Тележки : Элементы\_участка : permanent

$Parameters

состояние : (свободен, занят, загружен, перемещается, прибыл, ожидает) = свободен

$End

$Resource\_type Роботы : Элементы\_участка : permanent

$Parameters

состояние : (свободен, занят) = свободен

$End

$Resource\_type Станки : Элементы\_участка : permanent

$Parameters

состояние : (свободен, загружается, готов\_к\_обработке, работает,разгружается, закончил\_обработку) = свободен

время\_работы : real

$End

$Resource\_type Детали : permanent

$Parameters

номер : integer

положение : such\_as Система.положение = накопитель\_1

состояние : (хранится, транспортируется, обрабатывается, обработка\_закончна) = хранится

$End

**Primer.rss (Ресурсы):**

$Resources

Система\_1 : Система trace накопитель\_1

Накопитель\_1 : Накопители trace 1 накопитель\_1 \* 10

Накопитель\_2 : Накопители trace 2 накопитель\_2 \* \*

Робот\_1 : Роботы trace 1 накопитель\_1 \*

Робот\_2 : Роботы trace 2 тележка\_1\_к \*

Робот\_3 : Роботы trace 3 тележка\_2\_к \*

Тележка\_1 : Тележки trace 1 тележка\_1\_н \*

Тележка\_2 : Тележки trace 2 тележка\_2\_н \*

Станок\_1 : Станки trace 1 станок\_1 \* 28

Станок\_2 : Станки trace 2 станок\_2 \* 28

Станок\_3 : Станки trace 3 станок\_3 \* 15

Деталь\_1 : Детали 1 \* \*

Деталь\_2 : Детали 2 \* \*

Деталь\_3 : Детали 3 \* \*

Деталь\_4 : Детали 4 \* \*

Деталь\_5 : Детали 5 \* \*

Деталь\_6 : Детали 6 \* \*

Деталь\_7 : Детали 7 \* \*

Деталь\_8 : Детали 8 \* \*

Деталь\_9 : Детали 9 \* \*

Деталь\_10 : Детали 10 \* \*

$End

**Primer.opr (Операции):**

$Operations

Таймер\_ : Работа\_таймера

Работа\_на\_погрузке\_1 : Погрузка\_детали

Доставка\_детали\_1 : Доставка\_детали 1 тележка\_1\_н тележка\_1\_к

Доставка\_детали\_2 : Доставка\_детали 2 тележка\_2\_н тележка\_2\_к

Установка\_на\_станке\_1 : Установка\_на\_станке станок\_1 тележка\_1\_к

Установка\_на\_станке\_2 : Установка\_на\_станке станок\_2 тележка\_1\_к

Установка\_на\_станке\_3 : Установка\_на\_станке станок\_3 тележка\_2\_к

Возврат\_робота\_ : Возврат\_робота

Возврат\_тележки\_1 : Возврат\_тележки 1 тележка\_1\_к тележка\_1\_н

Возврат\_тележки\_2 : Возврат\_тележки 2 тележка\_2\_к тележка\_2\_н

Обработка\_на\_станке\_1 : Обработка\_на\_станке станок\_1

Обработка\_на\_станке\_2 : Обработка\_на\_станке станок\_2

Обработка\_на\_станке\_3 : Обработка\_на\_станке станок\_3

Разгрузка\_станков\_1 : Разгрузка\_станков станок\_1

Разгрузка\_станков\_2 : Разгрузка\_станков станок\_2

Окончание\_обработки\_ : Окончание\_обработки станок\_3

$End

**Primer.frm (Анимация):**

$Frame fram\_1

$Back\_picture = <127 127 127> 800 800

Show

text [10, 5, 50, 25, <127 127 127>, <100 255 0>, 'Время:' ]

text [60, 5, 150, 25, <127 127 127>, <100 255 0>, Time\_now ]

text [10,70 ,350 ,25 , <127 127 127>, <0 0 0>, 'Станок 1 в состоянии:' ]

text [350,70 ,350 ,25 , <127 127 127>, <0 0 0>, Станок\_1.состояние]

text [500,70 ,350 ,25 , <127 127 127>, <0 0 0>, Станок\_1.положение]

text [10,85 ,350 ,25 , <127 127 127>, <0 0 0>, 'Станок 2 в состоянии:' ]

text [350,85 ,350 ,25 , <127 127 127>, <0 0 0>, Станок\_2.состояние]

text [500,85 ,350 ,25 , <127 127 127>, <0 0 0>, Станок\_2.положение]

text [10,100 ,350 ,25 , <127 127 127>, <0 0 0>, 'Станок 3 в состоянии:' ]

text [350,100 ,350 ,25 , <127 127 127>, <0 0 0>, Станок\_3.состояние]

text [500,100 ,350 ,25 , <127 127 127>, <0 0 0>, Станок\_3.положение]

text [10,120 ,350 ,25 , <127 127 127>, <0 0 0>, 'Тележка\_1 в состоянии:' ]

text [350,120 ,350 ,25 , <127 127 127>, <0 0 0>, Тележка\_1.состояние]

text [500,120 ,350 ,25 , <127 127 127>, <0 0 0>, Тележка\_1.положение]

text [10,135 ,350 ,25 , <127 127 127>, <0 0 0>, 'Тележка\_2 в состоянии:' ]

text [350,135 ,350 ,25 , <127 127 127>, <0 0 0>, Тележка\_2.состояние]

text [500,135 ,350 ,25 , <127 127 127>, <0 0 0>, Тележка\_2.положение]

text [10,150 ,350 ,25 , <127 127 127>, <0 0 0>, 'Робот\_1 в состоянии:' ]

text [350,150 ,350 ,25 , <127 127 127>, <0 0 0>, Робот\_1.состояние]

text [500,150 ,350 ,25 , <127 127 127>, <0 0 0>, Робот\_1.положение]

text [10,165 ,350 ,25 , <127 127 127>, <0 0 0>, 'Робот\_2 в состоянии:' ]

text [350,165 ,350 ,25 , <127 127 127>, <0 0 0>, Робот\_2.состояние]

text [500,165 ,350 ,25 , <127 127 127>, <0 0 0>, Робот\_2.положение]

text [10,180 ,350 ,25 , <127 127 127>, <0 0 0>, 'Робот\_3 в состоянии:' ]

text [350,180 ,350 ,25 , <127 127 127>, <0 0 0>, Робот\_3.состояние]

text [500,180 ,350 ,25 , <127 127 127>, <0 0 0>, Робот\_3.положение]

text [10,200 ,350 ,25 , <127 127 127>, <0 0 0>, 'Кол-во детале в 1-м накопителе:' ]

text [350,200 ,350 ,25 , <127 127 127>, <0 0 0>, Накопитель\_1.текущее\_количество]

text [10,215 ,350 ,25 , <127 127 127>, <0 0 0>, 'Кол-во детале в 2-м накопителе:' ]

text [350,215 ,350 ,25 , <127 127 127>, <0 0 0>, Накопитель\_2.текущее\_количество]

text [10,300 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 1:' ]

text [250,300 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_1.положение]

text [350,300 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_1.состояние]

text [10,315 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 2:' ]

text [250,315 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_2.положение]

text [350,315 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_2.состояние]

text [10,330 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 3:' ]

text [250,330 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_3.положение]

text [350,330 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_3.состояние]

text [10,345 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 4:' ]

text [250,345 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_4.положение]

text [350,345 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_4.состояние]

text [10,360 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 5:' ]

text [250,360 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_5.положение]

text [350,360 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_5.состояние]

text [10,375 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 6:' ]

text [250,375 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_6.положение]

text [350,375 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_6.состояние]

text [10,390 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 7:' ]

text [250,390 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_7.положение]

text [350,390 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_7.состояние]

text [10,405 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 8:' ]

text [250,405 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_8.положение]

text [350,405 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_8.состояние]

text [10,420 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 9:' ]

text [250,420 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_9.положение]

text [350,420 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_9.состояние]

text [10,435 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 10:' ]

text [250,435 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_10.положение]

text [350,435 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь\_10.состояние]

$End

**Primer.fun (Константы, последовательности, функции):**

$Constant

Время\_погрузки : real = 1.5

Время\_установки\_на\_станок : real = 1.0

Время\_разгрузки\_станка : real = 0.5

Время\_возврата : real = 0.2

Время\_доставки : real = 2.0

$End

$Sequence Экспоненциальный : real

$Type = exponential 123456789

$End

$Function место\_возврата : such\_as Накопители.положение

$Type = algorithmic

$Parameters

текущее\_место : such\_as Накопители.положение

$Body

Calculate\_if текущее\_место = станок\_1 место\_возврата = тележка\_1\_к

Calculate\_if текущее\_место = станок\_2 место\_возврата = тележка\_1\_к

Calculate\_if текущее\_место = тележка\_2\_н место\_возврата = тележка\_1\_к

Calculate\_if текущее\_место = тележка\_1\_н место\_возврата = накопитель\_1

Calculate\_if текущее\_место = станок\_3 место\_возврата = тележка\_2\_к

Calculate\_if 0 = 0 место\_возврата = текущее\_место

$End

$Function робот\_станок : integer[0..2]

$Type = table

$Parameters

Номер\_робота\_ : integer [1..3]

Номер\_станка\_ : integer [1..3]

$Body

{Номер\_робота}

{ 1 2 3 }

{Но- 1} 0 1 0

{мер 2} 0 1 0

{станка 3} 0 0 1

$End

**Primer.smr (Прогон):**

Model\_name = Primer

Resource\_file = Primer

OprIev\_file = Primer

Statistic\_file = Primer

Results\_file = Primer

Trace\_file = Primer

Frame\_file = Primer

Frame\_number = 1

Show\_mode = Animation

Show\_rate = 5000.0

Terminate\_if Накопитель\_выходной.Количество\_деталей = 10

# Приложение 2. Полный синтаксический анализ описания типа данных (rdortp.y).

namespace rdoParse

{

%}

%start rtp\_list

%%

rtp\_list: /\* empty \*/

| rtp\_list rtp\_res\_type

| error {

PARSER->error( "Ожидается ключевое слово $Resource\_type" );

};

rtp\_res\_type: rtp\_header RDO\_Parameters rtp\_body RDO\_End

{

RDORTPResType\* res\_type = reinterpret\_cast<RDORTPResType\*>($1);

if ( res\_type->getParams().empty() )

{

PARSER->warning( @2, rdo::format( "Тип ресурса '%s' не содежит параметров", res\_type->name().c\_str() ) );

}

}

| rtp\_header RDO\_Parameters rtp\_body {

PARSER->error( @2, "Не найдено ключевое слово $End" );

}

| rtp\_header error {

PARSER->error( @2, "Не найдено ключевое слово $Parameters" );

};

rtp\_header: RDO\_Resource\_type RDO\_IDENTIF\_COLON rtp\_vid\_res

{

LEXER->m\_enum\_param\_cnt = 0;

RDOValue\* type\_name = reinterpret\_cast<RDOValue\*>($2);

std::string name = type\_name->value().getIdentificator();

const RDORTPResType\* \_rtp = PARSER->findRTPResType( name );

if ( \_rtp ) {

PARSER->error\_push\_only( type\_name->src\_info(), rdo::format("Тип ресурса уже существует: %s", name.c\_str()) );

PARSER->error\_push\_only( \_rtp->src\_info(), "См. первое определение" );

PARSER->error\_push\_done();

}

RDORTPResType\* rtp = new RDORTPResType( PARSER, type\_name->src\_info(), $3 != 0 );

$$ = (int)rtp;

}

| RDO\_Resource\_type RDO\_IDENTIF\_COLON RDO\_IDENTIF\_COLON rtp\_vid\_res

{

LEXER->m\_enum\_param\_cnt = 0;

RDOValue\* type\_name = reinterpret\_cast<RDOValue\*>($2);

std::string name = type\_name->value().getIdentificator();

const RDORTPResType\* \_rtp = PARSER->findRTPResType( name );

if ( \_rtp ) {

PARSER->error\_push\_only( type\_name->src\_info(), rdo::format("Тип ресурса уже существует: %s", name.c\_str()) );

PARSER->error\_push\_only( \_rtp->src\_info(), "См. первое определение" );

PARSER->error\_push\_done();

}

RDOValue\* prnt\_type\_name = reinterpret\_cast<RDOValue\*>($3);

std::string prnt\_name = prnt\_type\_name->value().getIdentificator();

const RDORTPResType\* \_rtp\_prnt = PARSER->findRTPResType( prnt\_name );

if ( \_rtp\_prnt ) {

RDORTPResType\* rtp = new RDORTPResType( PARSER, type\_name->src\_info(), $4 != 0 );

rsint t\_ind=0, col\_par=\_rtp\_prnt->getParams().size();

while (t\_ind<col\_par)

{

rtp->addParam(\_rtp\_prnt->getParams()[t\_ind]);

PARSER->warning( rdo::format("Параметр %s передан от родителя %s потомку %s", \_rtp\_prnt->getParams()[t\_ind]->src\_info().src\_text().c\_str(), prnt\_name.c\_str(),name.c\_str()) );

t\_ind=t\_ind+1;

}

$$ = (int)rtp;

PARSER->warning( rdo::format("Тип ресурса %s является потомком типа ресурса %s", name.c\_str(), prnt\_name.c\_str()) );

}

else {

PARSER->error\_push\_only( rdo::format("Родительский тип ресурса не существует: %s", prnt\_name.c\_str()) );

PARSER->error\_push\_done();

}

}

| RDO\_Resource\_type RDO\_IDENTIF\_COLON error {

PARSER->error( @2, "Не указан вид ресурса" );

}

| RDO\_Resource\_type RDO\_IDENTIF\_COLON RDO\_IDENTIF\_COLON error {

PARSER->error( @3, "Не указан вид ресурса" );

}

| RDO\_Resource\_type error {

std::string str( LEXER->YYText() );

PARSER->error( @2, rdo::format("Ошибка в описании имени типа ресурса: %s", str.c\_str()) );

};

rtp\_vid\_res: RDO\_permanent { $$ = 1; }

| RDO\_temporary { $$ = 0; };

rtp\_body: /\* empty \*/ {

}

| rtp\_body rtp\_param {

RDORTPParam\* param = reinterpret\_cast<RDORTPParam\*>($2);

PARSER->getLastRTPResType()->addParam( param );

};

rtp\_param: RDO\_IDENTIF\_COLON param\_type fuzzy\_terms\_list

{

RDOValue\* param\_name = reinterpret\_cast<RDOValue\*>($1);

RDORTPParamType\* param\_type = reinterpret\_cast<RDORTPParamType\*>($2);

RDORTPFuzzyTermsSet\* terms\_set = reinterpret\_cast<RDORTPFuzzyTermsSet\*>($3);

if ( terms\_set->empty() )

{

RDORTPParam\* param = new RDORTPParam( PARSER->getLastRTPResType(), param\_name->src\_info(), param\_type );

param\_type->reparent( param );

if ( param\_type->typeID() == rdoRuntime::RDOType::t\_enum ) {

static\_cast<RDORTPEnumParamType\*>(param\_type)->enum\_name = rdo::format( "%s.%s", PARSER->getLastRTPResType()->name().c\_str(), param\_name->src\_info().src\_text().c\_str() );

}

$$ = (int)param;

}

else

{

RDORTPFuzzyParam\* param = new RDORTPFuzzyParam( PARSER, param\_name->src\_info(), terms\_set );

param\_type->reparent( param );

$$ = (int)param;

}

}

| RDO\_IDENTIF\_COLON error {

if ( PARSER->lexer\_loc\_line() == @1.last\_line ) {

std::string str( LEXER->YYText() );

PARSER->error( @2, rdo::format( "Неверный тип параметра: %s", str.c\_str() ) );

} else {

PARSER->error( @1, "Ожидается тип параметра" );

}

}

| error {

PARSER->error( @1, "Неправильное описание параметра" );

};

fuzzy\_terms\_list: /\* empty \*/ {

RDORTPFuzzyTermsSet\* terms\_set = new RDORTPFuzzyTermsSet( PARSER );

$$ = (int)terms\_set;

}

| fuzzy\_terms\_list fuzzy\_term {

RDORTPFuzzyTermsSet\* terms\_set = reinterpret\_cast<RDORTPFuzzyTermsSet\*>($1);

RDORTPFuzzyTerm\* term = reinterpret\_cast<RDORTPFuzzyTerm\*>($2);

terms\_set->add( term );

$$ = $1;

};

fuzzy\_term: RDO\_Fuzzy\_Term RDO\_IDENTIF {

RDOValue\* param\_name = reinterpret\_cast<RDOValue\*>($2);

// RDORTPFuzzyMembershiftFun\* fuzzy\_membershift\_fun = reinterpret\_cast<RDORTPFuzzyMembershiftFun\*>($3);

// RDORTPFuzzyTerm\* fuzzy\_term = new RDORTPFuzzyTerm( PARSER, param\_name->src\_info(), fuzzy\_membershift\_fun );

// fuzzy\_membershift\_fun->reparent( fuzzy\_term );

// $$ = (int)fuzzy\_term;

};

fuzzy\_membershift\_fun: /\* empty \*/ {

RDORTPFuzzyMembershiftFun\* fun = new RDORTPFuzzyMembershiftFun( PARSER );

$$ = (int)fun;

}

| fuzzy\_membershift\_fun membershift\_point {

RDORTPFuzzyMembershiftFun\* fun = reinterpret\_cast<RDORTPFuzzyMembershiftFun\*>($1);

RDORTPFuzzyMembershiftPoint\* point = reinterpret\_cast<RDORTPFuzzyMembershiftPoint\*>($2);

fun->add( point );

$$ = $1;

//Задание функции принадлежности точками - вершинами ломанных кривых

};

membershift\_point: '(' RDO\_REAL\_CONST ',' RDO\_REAL\_CONST ')' {

double x\_value = reinterpret\_cast<RDOValue\*>($2)->value().getDouble();

double y\_value = reinterpret\_cast<RDOValue\*>($4)->value().getDouble();

RDORTPFuzzyMembershiftPoint\* fuzzy\_membershift\_point = new RDORTPFuzzyMembershiftPoint( PARSER, RDOParserSrcInfo( @1, @5 ), x\_value, y\_value);

$$ = (int)fuzzy\_membershift\_point;

}

| '(' RDO\_REAL\_CONST ',' RDO\_REAL\_CONST ')' ',' {

double x\_value = reinterpret\_cast<RDOValue\*>($2)->value().getDouble();

double y\_value = reinterpret\_cast<RDOValue\*>($4)->value().getDouble();

RDORTPFuzzyMembershiftPoint\* fuzzy\_membershift\_point = new RDORTPFuzzyMembershiftPoint( PARSER, RDOParserSrcInfo( @1, @5 ), x\_value, y\_value);

$$ = (int)fuzzy\_membershift\_point;

}

| '(' RDO\_REAL\_CONST ',' RDO\_INT\_CONST ')' {

double x\_value = reinterpret\_cast<RDOValue\*>($2)->value().getDouble();

double y\_value = reinterpret\_cast<RDOValue\*>($4)->value().getDouble();

RDORTPFuzzyMembershiftPoint\* fuzzy\_membershift\_point = new RDORTPFuzzyMembershiftPoint( PARSER, RDOParserSrcInfo( @1, @5 ), x\_value, y\_value);

$$ = (int)fuzzy\_membershift\_point;

}

| '(' RDO\_REAL\_CONST ',' RDO\_INT\_CONST ')' ',' {

double x\_value = reinterpret\_cast<RDOValue\*>($2)->value().getDouble();

double y\_value = reinterpret\_cast<RDOValue\*>($4)->value().getDouble();

RDORTPFuzzyMembershiftPoint\* fuzzy\_membershift\_point = new RDORTPFuzzyMembershiftPoint( PARSER, RDOParserSrcInfo( @1, @5 ), x\_value, y\_value);

$$ = (int)fuzzy\_membershift\_point;

};

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

// ---------- Описание типа параметра

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

param\_type: RDO\_integer param\_int\_diap param\_int\_default\_val

{

RDORTPIntDiap\* diap = reinterpret\_cast<RDORTPIntDiap\*>($2);

RDORTPDefVal\* dv = reinterpret\_cast<RDORTPDefVal\*>($3);

RDORTPIntParamType\* rp = new RDORTPIntParamType( PARSER->getLastParsingObject(), diap, dv, RDOParserSrcInfo( @1, @3 ) );

$$ = (int)rp;

}

| RDO\_real param\_real\_diap param\_real\_default\_val

{

RDORTPRealDiap\* diap = reinterpret\_cast<RDORTPRealDiap\*>($2);

RDORTPDefVal\* dv = reinterpret\_cast<RDORTPDefVal\*>($3);

RDORTPRealParamType\* rp = new RDORTPRealParamType( PARSER->getLastParsingObject(), diap, dv, RDOParserSrcInfo( @1, @3 ) );

$$ = (int)rp;

}

| RDO\_string param\_string\_default\_val

{

RDORTPDefVal\* dv = reinterpret\_cast<RDORTPDefVal\*>($2);

RDORTPStringParamType\* rp = new RDORTPStringParamType( PARSER->getLastParsingObject(), dv, RDOParserSrcInfo( @1, @2 ) );

$$ = (int)rp;

}

| RDO\_bool param\_bool\_default\_val

{

RDORTPDefVal\* dv = reinterpret\_cast<RDORTPDefVal\*>($2);

RDORTPBoolParamType\* rp = new RDORTPBoolParamType( PARSER->getLastParsingObject(), dv, RDOParserSrcInfo( @1, @2 ) );

$$ = (int)rp;

}

| param\_enum param\_enum\_default\_val

{

LEXER->m\_enum\_param\_cnt = 0;

RDORTPEnum\* enu = reinterpret\_cast<RDORTPEnum\*>($1);

RDORTPDefVal\* dv = reinterpret\_cast<RDORTPDefVal\*>($2);

if ( dv->isExist() )

{

enu->findEnumValueWithThrow( dv->value().src\_pos(), dv->value().value().getAsString() ); // Если не найдено, то будет сообщение об ошибке, т.е. throw

}

RDORTPEnumParamType\* rp = new RDORTPEnumParamType( PARSER->getLastParsingObject(), enu, dv, RDOParserSrcInfo( @1, @2 ) );

$$ = (int)rp;

}

| param\_such\_as

{

const RDORTPParam\* param = reinterpret\_cast<RDORTPParam\*>($1);

RDOParserSrcInfo src\_info( @1 );

src\_info.setSrcText( "such\_as " + (param->getResType() ? param->getResType()->name() + "." : "") + param->name() );

$$ = (int)param->getType()->constructorSuchAs( src\_info );

}

| param\_such\_as '=' RDO\_INT\_CONST

{

const RDORTPParam\* param = reinterpret\_cast<RDORTPParam\*>($1);

RDOParserSrcInfo src\_info( @1, @3 );

src\_info.setSrcText( "such\_as " + (param->getResType() ? param->getResType()->name() + "." : "") + param->name() );

$$ = (int)param->getType()->constructorSuchAs( src\_info, \*reinterpret\_cast<RDOValue\*>($3) );

}

| param\_such\_as '=' RDO\_REAL\_CONST

{

const RDORTPParam\* param = reinterpret\_cast<RDORTPParam\*>($1);

RDOParserSrcInfo src\_info( @1, @3 );

src\_info.setSrcText( "such\_as " + (param->getResType() ? param->getResType()->name() + "." : "") + param->name() );

$$ = (int)param->getType()->constructorSuchAs( src\_info, \*reinterpret\_cast<RDOValue\*>($3) );

}

| param\_such\_as '=' RDO\_IDENTIF

{

const RDORTPParam\* param = reinterpret\_cast<RDORTPParam\*>($1);

RDOParserSrcInfo src\_info( @1, @3 );

src\_info.setSrcText( "such\_as " + (param->getResType() ? param->getResType()->name() + "." : "") + param->name() );

$$ = (int)param->getType()->constructorSuchAs( src\_info, \*reinterpret\_cast<RDOValue\*>($3) );

}

| param\_such\_as '=' error

{

PARSER->error( "Ожидается зачение по-умолчанию" );

};

/\*

| RDO\_integer error {

PARSER->error( @2, "Ошибка после ключевого слова integer. Возможно, не хватает значения по-умолчанию." );

}

| RDO\_real error {

PARSER->error( @2, "Ошибка после ключевого слова real. Возможно, не хватает значения по-умолчанию." );

}

| param\_enum error {

PARSER->error( @2, "Ошибка после перечислимого типа. Возможно, не хватает значения по-умолчанию." );

};

\*/

param\_int\_diap: /\* empty \*/ {

YYLTYPE pos = @0;

pos.first\_line = pos.last\_line;

pos.first\_column = pos.last\_column;

RDORTPIntDiap\* diap = new RDORTPIntDiap( PARSER, pos );

$$ = (int)diap;

}

| '[' RDO\_INT\_CONST RDO\_dblpoint RDO\_INT\_CONST ']' {

RDORTPIntDiap\* diap = new RDORTPIntDiap( PARSER, reinterpret\_cast<RDOValue\*>($2)->value().getInt(), reinterpret\_cast<RDOValue\*>($4)->value().getInt(), RDOParserSrcInfo( @1, @5 ), @4 );

$$ = (int)diap;

}

| '[' RDO\_REAL\_CONST RDO\_dblpoint RDO\_REAL\_CONST {

PARSER->error( @2, "Требуется целочисленный диапазон, указан вещественный" );

}

| '[' RDO\_REAL\_CONST RDO\_dblpoint RDO\_INT\_CONST {

PARSER->error( @2, "Требуется целочисленный диапазон, указан вещественный" );

}

| '[' RDO\_INT\_CONST RDO\_dblpoint RDO\_REAL\_CONST {

PARSER->error( @4, "Требуется целочисленный диапазон, указан вещественный" );

}

| '[' RDO\_INT\_CONST RDO\_dblpoint RDO\_INT\_CONST error {

PARSER->error( @4, "Диапазон задан неверно" );

}

| '[' RDO\_INT\_CONST RDO\_dblpoint error {

PARSER->error( @4, "Диапазон задан неверно" );

}

| '[' error {

PARSER->error( @2, "Диапазон задан неверно" );

};

param\_real\_diap: /\* empty \*/ {

YYLTYPE pos = @0;

pos.first\_line = pos.last\_line;

pos.first\_column = pos.last\_column;

RDORTPRealDiap\* diap = new RDORTPRealDiap( PARSER, pos );

$$ = (int)diap;

}

| '[' RDO\_REAL\_CONST RDO\_dblpoint RDO\_REAL\_CONST ']' {

double min = reinterpret\_cast<RDOValue\*>($2)->value().getDouble();

double max = reinterpret\_cast<RDOValue\*>($4)->value().getDouble();

RDORTPRealDiap\* diap = new RDORTPRealDiap( PARSER, min, max, RDOParserSrcInfo( @1, @5 ), @4 );

$$ = (int)diap;

}

| '[' RDO\_REAL\_CONST RDO\_dblpoint RDO\_INT\_CONST ']' {

double min = reinterpret\_cast<RDOValue\*>($2)->value().getDouble();

double max = reinterpret\_cast<RDOValue\*>($4)->value().getDouble();

RDORTPRealDiap\* diap = new RDORTPRealDiap( PARSER, min, max, RDOParserSrcInfo( @1, @5 ), @4 );

$$ = (int)diap;

}

| '[' RDO\_INT\_CONST RDO\_dblpoint RDO\_REAL\_CONST ']' {

double min = reinterpret\_cast<RDOValue\*>($2)->value().getDouble();

double max = reinterpret\_cast<RDOValue\*>($4)->value().getDouble();

RDORTPRealDiap\* diap = new RDORTPRealDiap( PARSER, min, max, RDOParserSrcInfo( @1, @5 ), @4 );

$$ = (int)diap;

}

| '[' RDO\_INT\_CONST RDO\_dblpoint RDO\_INT\_CONST ']' {

double min = reinterpret\_cast<RDOValue\*>($2)->value().getDouble();

double max = reinterpret\_cast<RDOValue\*>($4)->value().getDouble();

RDORTPRealDiap\* diap = new RDORTPRealDiap( PARSER, min, max, RDOParserSrcInfo( @1, @5 ), @4 );

$$ = (int)diap;

}

| '[' RDO\_REAL\_CONST RDO\_dblpoint RDO\_REAL\_CONST error {

PARSER->error( @4, "Диапазон задан неверно" );

}

| '[' RDO\_REAL\_CONST RDO\_dblpoint RDO\_INT\_CONST error {

PARSER->error( @4, "Диапазон задан неверно" );

}

| '[' RDO\_INT\_CONST RDO\_dblpoint RDO\_REAL\_CONST error {

PARSER->error( @4, "Диапазон задан неверно" );

}

| '[' RDO\_INT\_CONST RDO\_dblpoint RDO\_INT\_CONST error {

PARSER->error( @4, "Диапазон задан неверно" );

}

| '[' RDO\_REAL\_CONST RDO\_dblpoint error {

PARSER->error( @4, "Диапазон задан неверно" );

}

| '[' RDO\_INT\_CONST RDO\_dblpoint error {

PARSER->error( @4, "Диапазон задан неверно" );

}

| '[' error {

PARSER->error( @2, "Диапазон задан неверно" );

};

param\_int\_default\_val: /\* empty \*/ {

$$ = (int)new RDORTPDefVal(PARSER);

}

| '=' RDO\_INT\_CONST {

$$ = (int)new RDORTPDefVal(PARSER, \*reinterpret\_cast<RDOValue\*>($2) );

}

| '=' RDO\_REAL\_CONST {

PARSER->error( @2, rdo::format("Целое число инициализируется вещественным: %f", reinterpret\_cast<RDOValue\*>($2)->value().getDouble()) );

}

| '=' error {

RDOParserSrcInfo \_src\_info(@1, @2, true);

if ( \_src\_info.src\_pos().point() )

{

PARSER->error( \_src\_info, "Не указано значение по-умолчанию для целого типа" );

}

else

{

PARSER->error( \_src\_info, "Неверное значение по-умолчанию для целого типа" );

}

};

param\_real\_default\_val: /\* empty \*/ {

$$ = (int)new RDORTPDefVal(PARSER);

}

| '=' RDO\_REAL\_CONST {

$$ = (int)new RDORTPDefVal(PARSER, \*reinterpret\_cast<RDOValue\*>($2));

}

| '=' RDO\_INT\_CONST {

$$ = (int)new RDORTPDefVal(PARSER, \*reinterpret\_cast<RDOValue\*>($2));

}

| '=' error {

RDOParserSrcInfo \_src\_info(@1, @2, true);

if ( \_src\_info.src\_pos().point() )

{

PARSER->error( \_src\_info, "Не указано значение по-умолчанию для вещественного типа" );

}

else

{

PARSER->error( \_src\_info, "Неверное значение по-умолчанию для вещественного типа" );

}

};

param\_string\_default\_val: /\* empty \*/

{

$$ = (int)new RDORTPDefVal(PARSER);

}

| '=' RDO\_STRING\_CONST

{

$$ = (int)new RDORTPDefVal(PARSER, \*reinterpret\_cast<RDOValue\*>($2));

}

| '=' error

{

RDOParserSrcInfo \_src\_info(@1, @2, true);

if ( \_src\_info.src\_pos().point() )

{

PARSER->error( \_src\_info, "Не указано значение по-умолчанию для строчного типа" );

}

else

{

PARSER->error( \_src\_info, "Неверное значение по-умолчанию для строчного типа" );

}

};

param\_bool\_default\_val: /\* empty \*/

{

$$ = (int)new RDORTPDefVal(PARSER);

}

| '=' RDO\_BOOL\_CONST

{

$$ = (int)new RDORTPDefVal(PARSER, \*reinterpret\_cast<RDOValue\*>($2));

}

| '=' error

{

RDOParserSrcInfo \_src\_info(@1, @2, true);

if ( \_src\_info.src\_pos().point() )

{

PARSER->error( \_src\_info, "Не указано значение по-умолчанию для булевского типа" );

}

else

{

PARSER->error( \_src\_info, "Неверное значение по-умолчанию для булевского типа" );

}

};

param\_enum: '(' param\_enum\_list ')' {

RDORTPEnum\* enu = reinterpret\_cast<RDORTPEnum\*>($2);

enu->setSrcPos( @1, @3 );

enu->setSrcText( enu->getEnums().asString() );

$$ = $2;

}

| '(' param\_enum\_list error {

PARSER->error( @2, "Перечисление должно заканчиваться скобкой" );

};

param\_enum\_list: RDO\_IDENTIF {

RDORTPEnum\* enu = new RDORTPEnum( PARSER->getLastParsingObject(), \*reinterpret\_cast<RDOValue\*>($1) );

enu->setSrcInfo( reinterpret\_cast<RDOValue\*>($1)->src\_info() );

LEXER->m\_enum\_param\_cnt = 1;

$$ = (int)enu;

}

| param\_enum\_list ',' RDO\_IDENTIF {

if ( LEXER->m\_enum\_param\_cnt >= 1 ) {

RDORTPEnum\* enu = reinterpret\_cast<RDORTPEnum\*>($1);

enu->add( \*reinterpret\_cast<RDOValue\*>($3) );

$$ = (int)enu;

} else {

PARSER->error( @3, "Ошибка в описании значений перечислимого типа" );

}

}

| param\_enum\_list RDO\_IDENTIF {

if ( LEXER->m\_enum\_param\_cnt >= 1 ) {

RDORTPEnum\* enu = reinterpret\_cast<RDORTPEnum\*>($1);

enu->add( \*reinterpret\_cast<RDOValue\*>($2) );

$$ = (int)enu;

PARSER->warning( @1, rdo::format("Пропущена запятая перед: %s", reinterpret\_cast<RDOValue\*>($2)->value().getIdentificator().c\_str()) );

} else {

PARSER->error( @2, "Ошибка в описании значений перечислимого типа" );

}

}

| param\_enum\_list ',' RDO\_INT\_CONST {

PARSER->error( @3, "Значение перечислимого типа не может быть цифрой" );

}

| param\_enum\_list ',' RDO\_REAL\_CONST {

PARSER->error( @3, "Значение перечислимого типа не может быть цифрой" );

}

| param\_enum\_list RDO\_INT\_CONST {

PARSER->error( @2, "Значение перечислимого типа не может быть цифрой" );

}

| param\_enum\_list RDO\_REAL\_CONST {

PARSER->error( @2, "Значение перечислимого типа не может быть цифрой" );

}

| RDO\_INT\_CONST {

PARSER->error( @1, "Значение перечислимого типа не может начинаться с цифры" );

}

| RDO\_REAL\_CONST {

PARSER->error( @1, "Значение перечислимого типа не может начинаться с цифры" );

};

param\_enum\_default\_val: /\* empty \*/ {

$$ = (int)new RDORTPDefVal(PARSER);

}

| '=' RDO\_IDENTIF {

$$ = (int)new RDORTPDefVal(PARSER, \*reinterpret\_cast<RDOValue\*>($2));

}

| '=' error {

RDOParserSrcInfo \_src\_info(@1, @2, true);

if ( \_src\_info.src\_pos().point() )

{

PARSER->error( \_src\_info, "Не указано значение по-умолчанию для перечислимого типа" );

}

else

{

PARSER->error( \_src\_info, "Неверное значение по-умолчанию для перечислимого типа" );

}

};

param\_such\_as: RDO\_such\_as RDO\_IDENTIF '.' RDO\_IDENTIF {

std::string type = reinterpret\_cast<RDOValue\*>($2)->value().getIdentificator();

std::string param = reinterpret\_cast<RDOValue\*>($4)->value().getIdentificator();

const RDORTPResType\* const rt = PARSER->findRTPResType( type );

if ( !rt ) {

PARSER->error( @2, rdo::format("Ссылка на неизвестный тип ресурса: %s", type.c\_str()) );

}

const RDORTPParam\* const rp = rt->findRTPParam( param );

if ( !rp ) {

PARSER->error( @4, rdo::format("Ссылка на неизвестный параметр ресурса: %s.%s", type.c\_str(), param.c\_str()) );

}

$$ = (int)rp;

}

| RDO\_such\_as RDO\_IDENTIF {

std::string constName = reinterpret\_cast<RDOValue\*>($2)->value().getIdentificator();

const RDOFUNConstant\* const cons = PARSER->findFUNConstant( constName );

if ( !cons ) {

PARSER->error( @2, rdo::format("Ссылка на несуществующую константу: %s", constName.c\_str()) );

}

$$ = (int)cons->getDescr();

}

| RDO\_such\_as RDO\_IDENTIF '.' error {

std::string type = reinterpret\_cast<RDOValue\*>($2)->value().getIdentificator();

const RDORTPResType\* const rt = PARSER->findRTPResType( type );

if ( !rt ) {

PARSER->error( @2, rdo::format("Ссылка на неизвестный тип ресурса: %s", type.c\_str()) );

} else {

PARSER->error( @4, "Ошибка при указании параметра" );

}

}

| RDO\_such\_as error {

PARSER->error( @2, "После ключевого слова such\_as необходимо указать тип и параметер ресурса для ссылки" );

};

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