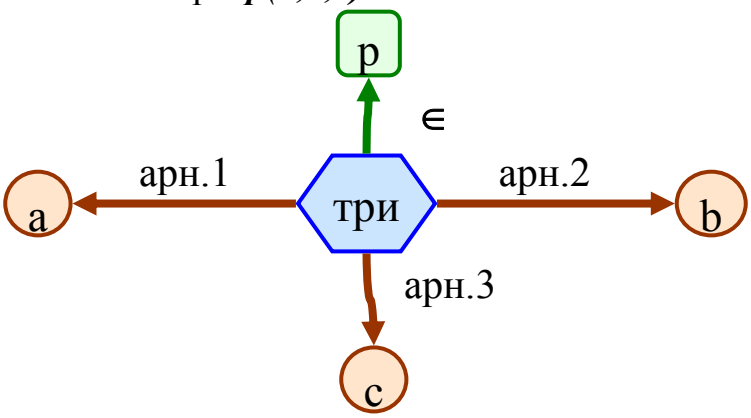
В. А. Валетов, Ю. П. Кузьмин,
А. А. Орлова, С. Д. Третьяков

ТЕХНОЛОГИЯ ПРИБОРОСТРОЕНИЯ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ СРС

Н-р: ***p(a)***

Н-р: ***p(a,b,c)***



Санкт-Петербург
2008

ИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНСТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ПРИОРИТЕТНЫЕ НАЦИОНАЛЬНЫЕ ПРОЕКТЫ «Образование»

ПОБЕДИТЕЛЬ КОНКУРСА ИННОВАЦИОННЫХ ОБРАЗОВАТЕЛЬНЫХ ПРОГРАММ ВУЗОВ

В.А.Валетов, Ю.П.Кузьмин, А.А.Орлова, С.Д.Третьяков

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ СРС

Учебно-методическое пособие

ИТМО
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Санкт- Петербург
2008

Валетов В.А. **МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ СРС** ./ , Кузьмин Ю.П., Орлова А.А., Третьяков С.Д. Учебное- методическое пособие .- СПб: СПбГУ ИТМО, 2008-32 с.

В пособии по выполнению СРС для обучающихся по магистерской программе « Технологическая подготовка производства приборов и систем» направление 200100-«Приборостроение» по дисциплине «Технология приборостроения» изложена методика самостоятельного освоения материала, который целесообразно использовать при решении технологических задач современными методами Пособие может быть полезным и для студентов всех образовательных программ направления «Приборостроение», в которых изучаются проблемы технологического характера, например, для студентов образовательных программ 200101 и 200107

«Рекомендовано УМО по образованию в области приборостроения и оптотехники.в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки 200100 – Приборостроение» Протокол № 3 от 29.04.08 заседания Президиума Совета УМО.

В 2007 году СПбГУ ИТМО стал победителем конкурса инновационных образовательных программ вузов России на 2007–2008 годы. Реализация инновационной образовательной программы «Инновационная система подготовки специалистов нового поколения в области информационных и оптических технологий» позволит выйти на качественно новый уровень подготовки выпускников и удовлетворить возрастающий спрос на специалистов в информационной, оптической и других высокотехнологичных отраслях экономики.

Оглавление

1.1. Рекомендации по использованию СРС

Как известно, часы на самостоятельную работу студентов (СРС) выделяются для:

- осознания, усвоения и расширения теоретического материала, излагаемого на лекциях;
- подготовки к выполнению и защите лабораторных и практических работ;
- выполнения домашних заданий, выступлений на семинарах, «круглых столах» и т.п.;
- для выполнения курсовых проектов и работ;
- для подготовки магистерской диссертации.

Все это и многое другое способствует формированию необходимых компетенций, изложенных в квалификационной характеристике по выше указанной программе.

Данное методическое пособие содержит руководство по выполнению курсового проекта по дисциплине «Технология приборостроения» и рекомендации по использованию СРС для формирования основных компетенций выпускника по данной магистерской программе.

Для получения новых знаний в области современных технологий необходимо в полной мере использовать Интернет и обязательно посещать российские и, особенно, международные выставки технологической направленности. Для глубокого освоения теоретических основ технологии рекомендуется ознакомление с публикациями технологического содержания независимо от года их выпуска. Чрезвычайно полезным является систематическое чтение периодической литературы по технологической тематике, включая иностранные журналы.

Для «преобразования» знаний в умение предусматривается выполнение курсовых проектов и работ, а также практических и лабораторных занятий. Практическое владение полученными знаниями (приобретение определенного опыта) должно формироваться в процессе технологических и преддипломных практик. Более прочные навыки по применению полученных знаний и умений лучше всего приобретать в процессе работы на промышленных предприятиях, поэтому на заключительной стадии освоения магистерской программы целесообразно совмещать учебу с работой на производственных предприятиях соответствующего профиля.

Современная магистерская подготовка предусматривает глубокие знания и практические навыки по использованию, совершенствованию и разработке новейших технологий, в том числе, компьютерных с использованием элементов искусственного интеллекта. Для приобретения таких знаний, умений и навыков, помимо проработки лекционных материалов, необходимо использовать современные компьютерные классы кафедры и университета, что позволяет не только решать реальные насущные технологические задачи, но и знакомиться через Интернет с новейшими достижениями за рубежом.

В настоящее время реальной стала необходимость освоения и широкого практического использования технологий быстрого прототипирования (Rapid Prototiping) и нанотехнологий. Теоретические основы этих знаний на кафедре ТПС даются в соответствующих разделах дисциплины «Технология приборостроения» уже в течение нескольких лет. В настоящее время кафедра получила возможность обеспечения практических навыков и определенного опыта в реальном использовании RP-технологий благодаря приобретению и введению в строй современной установки 3D-принтер. Так как такая установка в настоящее время является уникальной не только для наших высших технических вузов, но и для промышленных предприятий, кафедра организует использование данной установки как для пропаганды ее возможностей, так и для выполнения заказов конструкторских и технологических фирм. К этой реальной работе привлекаются аспиранты и студенты выпускного курса магистратуры.

Таким образом, кафедра обеспечивает выпуск компетентных специалистов высокой квалификации.

1.2. Рекомендации по изучению языка программирования CLIPS для разработки экспертных систем технологического назначения

Система, претендующая называться экспертной, должна обладать знаниями. Эти знания, естественно, должны быть ориентированы на конкретную предметную область, и из этих знаний должно непосредственно вытекать решение проблемы. Именно поэтому знания в экспертных системах предполагают определенную организацию и интеграцию (отдельные факты, сведения должны каким–либо образом соотноситься друг с другом и образовывать между собой определенные связи). То есть, знания должны быть соответствующе представлены.

В области экспертных систем представление знаний означает не что иное, как систематизированную методику описания на машинном уровне того, что знает человек–эксперт, специализирующийся в конкретной предметной области. Представление знаний должно позволять извлекать их в нужной ситуации с помощью относительно несложного и более–менее естественного механизма. Следует понимать, что недостаточно простого перевода знаний в форму, пригодную для хранения на машинных носителях. Чтобы достаточно быстро извлекать те элементы знаний, которые наиболее пригодны в конкретной ситуации, база знаний должна обладать достаточно развитыми средствами контекстной адресации и индексирования. Тогда программа, использующая знания, сможет управлять последовательностью применения отдельных "элементов" знания, даже не обладая точной информацией о том, как они хранятся.

Подобные средства предлагает разработчикам язык представления знаний CLIPS, позволяющий характеризовать представленные знания функционально, то есть в терминах действия, а не в терминах структурной

организации. Применение CLIPS для построения систем, основанных на знаниях, может быть обусловлено следующими причинами:

- этот язык является свободно распространяемым программным продуктом;
- его исполнительная система обладает вполне приемлемой производительностью;
- имеет четко сформулированный синтаксис;
- в него включено множество опробованных на практике конструкций из других инструментальных средств;
- язык допускает вызов внешних функций, написанных на других языках программирования; в свою очередь модули, написанные на CLIPS, могут быть вызваны программами, написанными на других языках;
- язык включает средства, позволяющие комбинировать порождающие правила и объектноориентированный подход.

CLIPS предлагает эвристические и процедурные подходы для представления знаний. Также средства CLIPS позволяют применять и объектноориентированный подход к организации знаний. Кроме того, язык предоставляет возможности комбинировать эти подходы.

Язык CLIPS (C Language Integrated Production System) начал разрабатываться в космическом центре Джонсона NASA в 1984 году. Сейчас CLIPS и документация на этот инструмент свободно распространяется через интернет (http://www.ghg.net/clips/CLIPS.html). Язык CLIPS свободен от недостатков предыдущих инструментальных средств для создания экспертных систем, основанных на языке LISP. Язык CLIPS получил большое распространение в государственных организациях и учебных заведениях, благодаря низкой стоимости, мощности, эффективности и переносимости с платформы на платформу. Например, даже Web-ориентированный инструментарий JESS (Java Expert System Shell), использующий язык представления знаний CLIPS, приобрел достаточную известность в настоящее время.

Следует отметить, что, несмотря на многочисленные преимущества функционального программирования, некоторые задачи лучше решать в терминах объектноориентированного программирования (ООП), для которого характерны три основные возможности: инкапсуляция (работа с классами), полиморфизм (работа с родовыми функциями, поддерживающими различное поведение функции в зависимости от типа аргументов), наследование (поддержка абстрактных классов). Объектноориентированное программирование поддерживает многие языки, в том числе Smalltalk, C++, Java, Common LISP Object System (CLOS). Язык CLIPS, в свою очередь, вобрал в себя основные преимущества C++ и CLOS.

Магистры могут познакомиться с языком CLIPS, получив через Интернет полный комплект документации на английском языке или прочитав специальную литературу на русском языке, представленную в конце данного пособия. В данном методическом пособии дается краткое неформальное введение в CLIPS, необходимое для программирования учебных задач.

Ниже рассмотрены предлагаемые CLIPS форматы представления данных и способы организации знаний.

Формат представления данных в CLIPS

В CLIPS предусмотрены три основных формата представления информации: факты, объекты и глобальные переменные.

Факты. Факты являются одной из основных форм высокого уровня для представления информации в системе CLIPS. Факт (fact) — это список элементарных значений, на которые ссылаются либо позиционно (упорядоченные (ordered) факты), либо по имени (неупорядоченные (non–ordered) или шаблонные (template) факты). Обращение к фактам осуществляется по индексу или адресу.

Каждый факт представляет часть информации и помещается в текущий список фактов (fact-list). Факты — фундаментальная единица данных, используемая правилами.

Факты могут быть добавлены в список фактов (используя команду assert), удалены из него (используя команду retract), изменены (используя команду modify) или скопированы (используя команду duplicate) в результате явного воздействия пользователя или при исполнении программы CLIPS. Число фактов в списке фактов и количество информации, которая может быть запомнена в факте, ограничено только объемом памяти компьютера. Если в список фактов заявлен факт, который точно соответствует уже имеющемуся там факту, эта вставка будет проигнорирована (впрочем, такое поведение может быть принудительно изменено).

Некоторые команды, такие как retract, modify и duplicate требуют наличия факта (что вполне логично: довольно затруднительно удалить, изменить или скопировать несуществующий факт!). Факт может быть указан или индексом (fact-index), или адресом (fact-address). Каждый раз при добавлении (или изменении) факта он получает уникальный целочисленный индекс, называемый fact-index. Индексы начинаются с нуля и увеличиваются на единицу для каждого нового или измененного факта. Каждый раз при выполнении команд reset (обновление рабочей памяти) или clear (очистка рабочей памяти) индексы фактов сбрасываются в ноль. Факт может быть указан и с использованием адреса факта (fact-address). Адрес факта может быть получен, перехватив возвращаемое значение команд, которые возвращают адреса (например, assert, modify, duplicate), или присвоением переменной адреса факта, который соответствует образцу в левой части правила (LHS — Left–Hand Side, т.е. список условий), например, так:

;;присвоение переменной адреса факта (somefact exists)

?somefact <- (somefact exists)

Идентификатор факта (fact identifier) представляет собой краткую нотацию для отображения факта. Формат идентификатора факта — *f–<fact-index>*, например, запись f–10 относится к факту с индексом 10.

Как отмечалось выше, факты хранятся в одном из двух форматов: упорядоченном (ordered) или неупорядоченном (non-ordered).

Упорядоченные факты состоят из символьного обозначения с последовательностью нуля или более полей, разделенных пробелами, ограниченного начальной круглой скобкой с левой стороны и завершающей круглой скобкой справа. Первое поле упорядоченного факта обозначает "отношение", которое следует применять к следующим полям в факте. Например, факт (father-of Jack Bill) означает, что Билл — отец Джека (Bill is father of Jack)Ниже приведены примеры упорядоченных фактов:

;; скорость - 80 км/ч ;
(speed is 80 km/h);
;; список бакалеи - хлеб молоко масло;
(grocery-list bread milk butter);
;; Робот находится в комнате;
(at room robot) .

Следующие символьные обозначения зарезервированы и не могут быть использованы в качестве первого поля в любых фактах: test, and, or, not, declare, logical, object, exists и проч. Эти слова зарезервированы и не могут быть использованы в качестве имен шаблонов в конструкциях deftemplate, но могут быть использованы в качестве имен слотов (slot) (см. в библиотеке), хотя это и не желательно.

Неупорядоченные факты. Упорядоченные факты кодируют информацию позиционно. Чтобы обратиться к этой информации, пользователь должен знать не только то, какие данные хранятся в факте, но и какое именно поле содержит те или иные конкретные сведения. Неупорядоченные (или deftemplate) факты обеспечивают пользователю способность абстрагироваться от структуры факта, назначая имя каждому полю факта. Конструкция deftemplate (пример см. в библиотеке) используется для создания шаблона, который затем может быть применен для получения доступа к полям шаблонных фактов по их имени. Конструкция deftemplate является аналогом определения записей или структур в таких языках программирования, как Pascal и C.:

(deftemplate имя_шаблона;
(slot атрибут1 (тип)[(default значение_по_умолчанию)];
(slot атрибут2 (тип)[(default значение_по_умолчанию)]);
……..
(slot атрибутN (тип)[(default значение_по_умолчанию)]).

*)*Эта конструкция позволяет определять шаблон с нулем или более поименованных полей (named fields) или заполнителей–слотов (slots). В отличие от упорядоченных фактов, для слотов шаблонного факта обязательно указание типа и значения. Кроме того, для слотов могут быть заданы значения по умолчанию. В неупорядоченных фактах отсутсвует ограничение на порядок следования полей, главное, чтобы было указано имя поля. Следует отметить, что слоты не могут быть использованы в упорядоченных фактах.

Шаблонные факты отличаются от упорядоченных прежде всего первым полем в составе факта. Первое поле всех фактов обязательно должно быть символьным обозначением, однако если это символьное обозначение относится к имени шаблона, значит этот факт — шаблонный. Как и в случае упорядоченных фактов, шаблонные факты заключены в круглые скобки.

Ниже приведены примеры шаблонных фактов:

(patient (name "Ivanov Ivan") (age 46));
(class (teacher "Alexey Alexeev") (N_pupils 28) (Room "37A"));
(car (model "BMW-Z3") (color "silver") (price 50000)).

Заметьте, что порядок слотов в шаблонном факте не важен. Например, следующие факты идентичны:

(patient (age 46) (name "Ivanov Ivan"));
(patient (name "Ivanov Ivan") (age 46));
……..
(car (color "silver") price 50000 (model "BMW-Z3"));
(car (price 50000) (model "BMW-Z3") (color "silver")).

В отличие от приведенных фактов, следующие упорядоченные факты не идентичны:

(class "Alexey Alexeev" 28 "37A";)
(class 28 "37A" "Alexey Alexeev");
(class "37A" "Alexey Alexeev" 28.

К шаблонным фактам также можно применять команды вставки, удаления, изменения и копирования. Изменение неупорядоченного факта приводит к изменению набора изменяемых слотов. Дублирование факта создает новый факт, идентичный оригинальному факту, а затем изменяется набор указанных слотов в пределах нового факта. Преимущество использования команд изменения и дублирования для неупорядоченных фактов состоит в том, что не требуется указывать те слоты, которые не подлежат изменению или копированию.

Факты инициализации (Initial Facts). Конструкция deffacts позволяет задать набор априорных или инициализирующих знаний. Когда командой reset обновляется рабочая память CLIPS, каждый факт, задаваемый конструкцией deffacts в базе знаний CLIPS, добавляется в список фактов.

Команда reset создает в базе знаний специальный инициализирующий факт — initial-fact. Используя его, можно создать правило, которое бы выполняло некоторые действия при запуске кода CLIPS. Поскольку условием, активизирующим стартовое правило, будет наличие в базе знаний факта инициализации (а он там всегда присутствует самым первым), это правило будет активизировано в первую очередь. Например, напишем следующее правило и сохраним в файле example.clp:

(defrule start;
(initial-fact);
=>
(printout t "Hello, world!" crlf).
)

Тогда в диалоговом окне CLIPS при загрузке этого кода будем наблюдать:

CLIPS> (load example.clp);
*
TRUE;
CLIPS> (reset);
==> f-0 (initial-fact);
==> Activation 0 start: f-0;
CLIPS> (run);
FIRE 1 start: f-0;
Hello, world!;
CLIPS>.

Глобальные переменные. Конструкция defglobal позволяет описывать переменные, которые являются глобальными в контексте окружения CLIPS. То есть, глобальная переменная доступна в любом месте окружения CLIPS и сохраняет свое значение независимо от других конструкций. Напротив, некоторые конструкции (как, например, defrule или deffunction) могут иметь собственные локальные переменные, которые задаются в пределах описания конструкции. Обращение к таким переменным возможно только изнутри конструкции, где они описаны; за ее пределами они не имеют значения. Глобальные переменные CLIPS подобны глобальным в процедурных языках программирования, таких как LISP или C. Однако, в отличие от того же C, глобальные переменные CLIPS слабо типизированы (отсутствуют ограничения на хранения значений различных простых типов данных).

Глобальные переменные могут быть доступны как часть процесса сопоставления образцов (pattern-match), однако их изменение не вызывает процесс сопоставления образцов. Функция bind используется, чтобы задать значения глобальным переменным. Значения глобальных переменных сбрасываются к начальным установочным значениям при выполнении команды окружения reset или если для глобальной переменной вызвана функция bind без соответствующего значения. Такой порядок может быть изменен, если использовать функцию set-reset-globals. Глобальные переменные могут быть удалены командами clear или undefglobal. Если элемент глобальных переменных отслеживается при трассировке, то соответствующее информационное сообщение будет отображаться каждый раз, когда значение глобальной переменной изменяется.

Синтаксис описания:

(defglobal [<defmodule-name>] ?<имя_глобальной_переменной>* = <выражение>)*

Здесь выражение в угловых скобках, как общепринято, обозначает отдельную сущность поименованного класса элементов, задаваемую пользователем.

В целом возможны и множественные конструкции defglobal. Любое число глобальных переменных может быть задано в описании defglobal.

Необязательный <defmodule-name> указывает модуль, в котором конструкция defglobals будет определена. Если ничего не указано, глобальные переменные будут размещены в текущем модуле. Если переменная была определена в предыдущей конструкции defglobal, её значение будет заменено значением, заданном в новой конструкции defglobal. Если при описании конструкции defglobal допущена ошибка, любые описания глобальной переменной, заданные до того, как ошибка обнаружена, будут оставаться в силе.

Команды, которые оперируют с глобальными переменными, как например ppdefglobal и undefglobal, требуют наличия символического имени глобальной переменной без "звездочек" (этом случае, например, используйте обозначение max, когда хотите сослаться на глобальную переменную ?*max*).

Глобальные переменные могут быть использованы в любом месте программы также, как и локальные переменные (за исключением 2–х случаев):

1) глобальные переменные не могут быть использованы в качестве параметрических переменных в конструкциях deffuncion, defmethod или message-handler;

2) глобальные переменные не могут быть использованы так же, как локальные переменные используются в списках условий правил для связывания их со значениями.

Таким образом, следующее правило неверное:

```
(defrule example
(fact ?*x*)
=>
)
```

А следующее правило корректно:

```
(defrule example;
(fact ?y&:(> ?y ?*x*)).
=>
)
```

Отметьте, что это правило не обязательно будет активизировано, когда значение ?*x* изменится. Например, если ?*x* составляет 4 и факт (fact 3) добавлен в рабочую память, условия правила не будут удовлетворены. Если же значение ?*x* изменится на 2, правило все равно не будет активизировано.

Объекты. Объекты в CLIPS могут быть описаны как символьные, строковые, целые или вещественные числа, значения с множеством полей, внешние адреса или объекты определенного пользователем класса. При описании объектов выделяют 2 основные части: свойства и их поведение. Класс — это шаблон общих свойств и поведения объектов данного класса.

Для создания пользовательского класса используется конструкция defclass. Объект пользовательского класса создается посредством функции

make-instance, и к созданному таким образом объекту можно обращаться по уникальному адресу. В пределах модульного контекста к объекту можно уникально обращаться и по имени. Ниже приведен пример создания простого класса и его экземпляра:

```
;; описание класса car (машина);
(defclass car;
(is-a user) ;; пользовательский класс;
(name) ;; наименование модели;
(made-by) ;; производитель;
;; описание экземпляра класса car;
(make-instance corvette of car;
(name lacetti);
(made-by chevrolet.)
)
```

Объекты в CLIPS разделяются на две важные категории: объекты простых типов и объекты пользовательских классов. Эти два вида объектов отличаются способом обращения к ним, создания и удаления, а также тем, как задаются их свойства.

К объектам простых типов обращаются, подставляя их значения, и они создаются и удаляются исключительно окружением CLIPS по мере необходимости. Объекты простых типов не имеют имен и слотов-заполнителей, и их классы предопределены CLIPS. Поведение объектов простых типов такое же, как и у объектов пользовательских классов. Пользователь может создавать собственные обработчики сообщений (которыми и оперируют классы в CLIPS — обычно над слотами объекта выполняются какие–то действия при получении объектом тех или иных сообщений) и присоединить их к объектам простых типов. Тем не менее, предполагается, что простые типы не будут часто использоваться в контексте объектно–ориентированного программирования в CLIPS.

Ниже представлены несколько примеров объектов и их классов:Объект (печатное представление) Класс

```
Rolls–RoyceSYMBOL
"Rolls–Royce"      STRING
8.0    FLOAT
8      INTEGER
(8.0 Rolls-Royce 8 [Rolls-Royce])      MULTIFIELD
<Pointer- 00CF61AB>   EXTERNAL-ADDRESS
[Rolls-Royce]      CAR (пользовательский класс)
```

На объект пользовательского класса ссылаются по имени или адресу, и они создаются и удаляются явно через сообщения и специальные функции. Свойства объекта пользовательского класса определяются набором слотов, которые объект получает от его класса. Как ранее упоминалось, слоты — это поименованные одиночные поля или мультиполя (multifields). Например, объект Rolls–Royce — это объект класса CAR. Одним из слотов в классе CAR

может быть "цена" (price), и значение этого слота для объекта Rolls–Royce составило бы, например, $75 000. Поведение объекта указано в терминах процедурного кода называемого обработчики сообщений (message-handlers), которые присоединены к классу объекта. Все объекты пользовательского класса имеют одинаковый набор слотов, но каждый объект может иметь различные значения для этих слотов. Однако два объекта, которые имеют одинаковый набор слотов, не обязательно принадлежат к одному и тому же классу, так как два различных класса могут иметь идентичные наборы слотов.

Базовая разница между слотами объекта и шаблона (неупорядоченного факта) состоит в наследовании. Наследование позволяет описывать свойства и поведение класса описывать в терминах других классов. COOL (CLIPS Object–Oriented Language — Объектно–ориентированный язык CLIPS) поддерживает множественное наследование: класс может напрямую наследовать слоты и обработчики сообщений от более чем одного класса. Так как наследование полезно только для слотов и обработчиков сообщений, часто не значимо наследование от одного из классов простых типов, как, например, MULTIFIELD или NUMBER. Это связано с тем, что эти классы не могут иметь слотов и обычно не имеют обработчиков сообщений.

Пользовательские классы могут быть конкретными и абстрактными. Абстрактные классы играют ту же роль, что и виртуальные классы в C++, то есть они используются только для порождения производных классов. Например, если имеем абстрактный класс PERSON, то используя механизм наследования можно создать производные классы WOMAN и MAN.

Использование объектно–ориентированных средств в CLIPS позволяет значительно упростить программирование правил, поскольку для обновления данных можно применять механизм передачи и обработки сообщений методами классов. Ниже представлен пример описания класса и его экземпляра из [1]:

;; определяем класс pistol, в котором будут перечислены свойств;,

;; необходимые для моделирования работы полуавтоматического пистолета;

(defclass pistol;

;; системные слоты;

(is-a USER) ;; pistol - это пользовательский класс ;

(role concrete) ;; это конкретный класс, т.е. возможно создание экземпляров этого класса;

(pattern-match reactive);; экземпляры класса pistol могут быть использованы в качестве объектов данных;

;; кот. можно сопоставлять с условиями в правилах и использовать в действи;,

;; определенных правилами;

(slot safety (type SYMBOL) (create-accessor read-write)) ;; предохраитель (on/off;)

(slot slide (type SYMBOL) (create-accessor read-write)) ;; затвор (forward/back);

(slot hammer (type SYMBOL) (create-accessor read-write)) ;; курок (back/down) ;

(slot chamber (type INTEGER) (create-accessor read-write)) ;; патронник (1/0) ;

(slot magazine (type SYMBOL) (create-accessor read-write)) ;; обойма (in/out) ;

(slot rounds (type INTEGER) (create-accessor read-write)) ;; патроны (текущее количество в обойме);

;; фацет create-accessor разрешает записывать в слот новое значение и считывать текущее;

)

```
;; формируем экземпляр класса pistol
(definstances pistols ;
 (PPK of pistol ;
 (safety on) ;; PPK установлен на предохранитель;
 (slide forward);; затвор в переднем положении ;
 (hammer down) ;; курок опущен;
 (chamber 0) ;; патронник пуст;
 (magazine out) ;; обойма вынута;
 (rounds 6) ;; в обойме 6 патронов;
 )
;; создаем новый класс pistol2 на базе класса pistol;
(deffclass pistol2;
 (is-a pistol;)
 (role concrete).
 ...
```

Объекты инициализации. Конструкция definstances позволяет задать априорные (инициализирующие) знания в виде набора экземпляров пользовательских классов.

При перезагрузке рабочей памяти CLIPS (командой reset) каждый экземпляр, определенный внутри конструкции definstances, добавляется в список экземпляров базы знаний CLIPS.

Механизмы представления знаний. CLIPS предоставляет три механизма представления знаний: эвристический, процедурный и объектно ориентированный. Рассмотрим эти механизмы подробнее.

Эвристический подход. Одним из основных подходов к представлению знаний в CLIPS является использование правил. Правила используются для представления эвристик ("правил влияния"), определяющих набор действий, которые необходимо выполнить в данной ситуации. Разработчик экспертной системы (инженер по знаниям) задает набор правил, которые совместно работают над разрешением проблемы. Правило (rule) состоит из антецедента (antecedent) и консеквента (consequent). Антецедент правила есть не что иное как часть "ЕСЛИ..." (то

есть список условий) и называется LHS (см. выше). Консеквент — это часть "ТО..." (то есть список действий) и называется RHS (Right Hand Side — правая часть правила).

Антецедент определяет набор условий, которые должны быть удовлетворены для активации правила. В CLIPS удовлетворение условий базируется на существовании или несуществовании определенных фактов в списке фактов (см. выше) или определенных экземпляров пользовательских классов в списке экземпляров базы знаний. Один вид условия, которое может быть задано, — это образец pattern. Образцы состоят из набора ограничений, которые используются, чтобы определить, какие факты или объекты удовлетворяют условию, заданное образцом. Процесс сопоставления фактов и объектов с образцами называется pattern-matching. CLIPS обладает механизмом машины логического вывода, который автоматически ставит в соответствие образцам текущее состояние списка фактов и списка экземпляров и определяет, какие правила активизировать.

Консеквент правила — это набор действий, которые выполняются в случае активизации правила. Действия соответствующих правил выполняются, когда машина логического вывода CLIPS проинструктирована начать выполнение соответствующих правил. Если возникла ситуация, что может быть активизировано более чем одно правило, машина логического вывода задействует стратегию разрешения конфликтов чтобы выбрать, какое именно правило должно быть активизировано. Действия выбранного правила выполняются (что может повлиять на список правил, которые могут быть активизированы в дальнейшем), затем машина логического вывода другое правило и его действия выполняются. Этот процесс выполняется до тех пор, пока существуют правила, которые могут быть активизированы либо пока программа не будет остановлена принудительно (прекращение выполнения программы может быть одним из действий консеквента правила).

Правила в CLIPS реализованы в привычной форме:

ЕСЛИ условие_1 и ... и условие_M удовлетворяются;
ТО;
ВЫПОЛНИТЬ действие_1 и ... и действие_N.

Следует заметить, что количество условных предпосылок M и число действий N, подлежащих выполнению в случае удовлетворения условий, в общем случае не равны. Если КАЖДОЕ условие в LHS находит себя среди фактов, то происходит активизация правила и выполнение ВСЕХ действий, записанных в его RHS. В противном случае правило не активизируется.

Для создания правила используется конструкция defrule, которая имеет следующий синтаксис:

```
(defrule имя_правила;
        [необязательный комментарий]
        [необязательное объявление];
        (условие_1);
        (условие_2);
        . . . . . .
```

```
(условие_M);
=>
(действие_1);
(действие_2);
. . . . . .
(действие_N)).
```

Обратите внимание, что LHS отделяется от RHS комбинацией символов "=>". Чтобы было более понятно, рассмотрим простенький пример с несколькими правилами.

```
;; описание шаблонов

;; fact - субъект с некоторым свойством
 (deftemplate fact
 (field subject (type SYMBOL)) ;; субъект
 (field property (type SYMBOL)) ;; его свойство
 )

;; action - действие, которое нужно выполнить
 (deftemplate action
 (field to_do (type STRING)) ;; что делать
 )

;; факты, задающие исходное состояние проблемы
 (deffacts basic_state
 (fact (subject day)(property day_off)) ;; выходной день
 (fact (subject weather)(property sunny));; погода солнечная
 )

;; правила
;; если день будний - нужно идти на работу
 (defrule work
 (fact (subject day)(property weekday))
 =>
 (assert (action (to_do "go to work")))
 (printout t "Нужно идти на работу!" crlf)
 )
;; если день выходной - можно остаться дома
 (defrule rest
 (fact (subject day)(property day_off))
 =>
 (assert (action (to_do "stay at home")))
 (printout t "Можно остаться дома!" crlf)
 )
```

```
;; если день выходной и к тому же нет дождя - можно пойти на прогулку
(defrule good_rest
(fact (subject day)(property day_off))
(fact (subject weather)(property ?P&~rainy))
=>
(assert (action (to_do "go for a walk")))
(printout t "Можно пойти на прогулку!" crlf)
)
;; если нужно куда-то идти и погода дождливая - взять зонт
(defrule take_umbrella
(action (to_do ?A))
(fact (subject weather)(property rainy))
=>
(if (eq (sub-string 1 2 ?A) "go") then (printout t "Нужно взять зонт!" crlf))
)

;; если нужно куда-то идти и погода солнечная - взять солнцезащитные очки
(defrule take_sunglasses
(action (to_do ?A))
(fact (subject weather)(property sunny))
=>
(if (eq (sub-string 1 2 ?A) "go") then (printout t "Нужно взять солнцезащитные очки!" crlf))
)
```

Программа состоит из трех частей: базы данных basic_state, описываемой конструкцией deffacts; шаблонов фактов fact и action; пяти правил work, rest, good_rest, take_umbrella, take_sunglasses.

В базе данных содержаться сведения о предметной области: какой сегодня день (будний или выходной) и какая погода (в данном примере не учитываются все разновидности погоды, а только знания о том, солнечная погода или дождливая). Сведения представлены в виде набора шаблонных фактов. Поле subject первого факта указывает на то, что речь идет о дне (day); поле property содержит значение свойства дня — выходной (day_off). Поле subject второго факта указывает на то, что речь идет о погоде (weather); поле property содержит значение свойства погоды — солнечная (sunny). Правило work определяет, нужно ли идти на работу. Правило активизируется, если будет выполнено условие "день будний", то есть если в базе данных будет факт (fact (subject day)(property weekday)). В этом случае в базу данных будет добавлен факт (action (to_do "go to work")), а также на экран будет выведено соответствующее сообщение командой printout.

Правила rest и good_rest определяют альтернативные действия в случае, если день — выходной. Первое правило активизируется единственным

условием в списке LHS (достаточно, чтобы в базе данных был факт (fact (subject day)(property day_off))), в то время как второе правило активизируется, если будут удовлетворены два условия в списке LHS, то есть если в базе данных будут факты, свидетельствующие, во–первых, что день выходной, а во–вторых — что погода не дождливая. В нашем примере очевидно будут активизированы оба эти правила, налицо конфликт — чему отдать предпочтение: отдыху дома или прогулке? Выбор приоритетного правила будет зависеть от стратегии разрешения конфликтов. Можно воспользоваться предусмотренным CLIPS свойством выпуклости и добавить, например, в правило rest после имени правила строку (declare (salience 10)), придав ему выпуклость 10, что сделает это правило приоритетным по сравнению с good_rest со значением выпуклости, равным 0 по умолчанию. Однако было бы предпочтительнее усовершенствовать программу, разделив контексты, в которых используются эти правила. Например, в контексте_1 (для человека ленивого) приоритетным действительно было бы остаться дома (активизация правила rest), а в контексте_2 (для человека активного) приоритетной была бы прогулка (активизация правила good_rest). Каждый контекст можно было бы обрабатывать отдельно.

Процедурный подход. CLIPS также поддерживает процедурный механизм, как большинство традиционных языков программирования, таких как Pascal или C. Функции, заданные конструкцией deffunction и родовые функции позволяют пользователю создавать новые исполнимые элементы, выполняющие полезные второстепенные действия или возвращают некоторое полезное значение. Созданные функции вызываются так же, как и встроенные функции CLIPS. Обработчики сообщений позволяют задать соответствующее поведение объекта в ответ на получаемые им сообщения. Функции, родовые функции и обработчики сообщений — это все процедурные части пользовательского кода, выполняемого CLIPS в соответствующих ситуациях.

В CLIPS процедурный и эвристический механизмы представления знаний могут тесно взаимодействовать путем вызова пользовательских функций как из LHS, так и из RHS. для создания пользовательских функций используется конструктор deffunction, который имеет следующий синтаксис:

(deffunction имя_функции;
[необязательный комментарий];
(список формальных параметров);
(действие_1;
(действие_2);
. . . . . .
(действие_N)).

Например, определим функцию om(x,y), которая возвращает целую часть частного от деления переменной y на переменную x:

(deffunction om;
(?x ?y);

(div ?y ?x)).

Обратите внимание, что в CLIPS имя переменной начинается с символа "?", что для вызова функции, в данном случае встроенной функции деления нацело div, используется префиксная нотация и что вся конструкция представляет собой список, состоящий из четырех полей. Так что у CLIPS ноги растут не только из C (CLIPS — C Language Integrated Production System), но и из LISP.

Предыдущий пример с планированием действий на день в зависимости от погоды и того, будний день или выходной, можно было бы усовершенствовать, используя процедурный механизм. Ведь на практике нам часто нужно спланировать свои действия на будущее, располагая конкретной датой, и не всегда мы достоверно знаем, будний это день или выходной. Ввиду этого полезной была бы функция, которая, оперируя с указанной датой, возвращала бы в качестве результата одно из двух значений — "будний день" или "выходной день".

Объектно-ориентированный подход. В CLIPS для организации объектно-ориентированного подхода к представлению знаний предусмотрен собственный объектно-ориентированный язык (функциональные возможности которого, тем не менее, уже, чем у того же C++) COOL — CLIPS Object Oriented Language. Язык COOL, включенный в состав CLIPS, имеет 17 системных классов, причем некоторые из них выполняют функции метаклассов. На самом верхнем уровне иерархии расположен класс Object, ниже которого в иерархии два производных класса — Primitive и User. Производными от Primitive являются классы Number, Instance, Address, Multifield и Lexeme. Производный класс от User — Initial-Object.

Все классы, определенные пользователем, являются производными от User, который отчасти выполняет функции метакласса. В нем реализованы практически все базовые обработчики инициализации и удаления объектов. Однако User все–таки не является метаклассом, поскольку классы, определенные пользователем, — это производные от User, а не его экземпляры. Initial-Object является экземпляром по умолчанию, который создается при выполнении функции definstances. Класс Primitive и его подклассы реализуют основные структуры данных — числа, символы строки, адреса и многокомпонентные объекты. Все вышеперечисленные классы, кроме Initial-Object, являются абстрактными и служат для определения родовых операций и структур данных.

Отличительной особенностью CLIPS являются конструкторы для создания баз знаний (БЗ):

defrule определение правил;
deffacts определение фактов;
deftemplate определение шаблона факта;
defglobal определение глобальных переменных;
deffunction определение функций;
defmodule определение модулей (совокупности правил);
defclass определение классов;

definstances определение объектов по шаблону, заданному defclass;
defmessagehandler определение сообщений для объектов;
defgeneric создание заголовка родовой функции;
defmethod определение метода родовой функции.

Конструкторы не возвращают никаких значений, в отличии от функций, например:

```
(deftemplate person
(slot name)
(slot age)
(multislot friends))
(deffacts people
(person (name Joe) (age 20))
(person (name Bob) (age 20))
(person (name Joe) (age 34))
(person (name Sue) (age 34))
(person (name Sue) (age 20)))
```

Пример функции:

```
(deffunction factorial (?a)
 (if (or (not (integerp ? a)) (< ? a0)) then
 (printout t "Factorial Error!" crlf)
 else
 (if (= ? a0) then
 1
 else
 (*? a (factorial ($-$ ? a1))))))
```

Правила в CLIPS состоят из предпосылок и следствия. Предпосылки также называют ЕСЛИ-частью правила, левой частью правила или LHS правила (left-hand side of rule). Следствие называют ТО-частью правила, правой частью правила или RHS правила (right-hand side of rule).

Пример правила представлен ниже:

```
(deftemplate data (slot x) (slot y))
(defrule twice
(data (x ? x) (y =(*2 ? x)))
)
(assert (data (x2) (y4)); f-0
 (data (x3) (y9))); f-1
```

Здесь самая распространенная в CLIPS функция assert добавляет новые факты в список правил. В противоположность assert функция retract удаляет факты из списка фактов, например:

```
(defrule vis11
 ?doors < — (fit ? wdfit)
 (test (eq ? wdfit no))
```

(assert (EVIDENCE OF MAJOR ACCIDENT))
(retract ? doors))

В этом правиле проверяется наличие факта doors и в случае его отсутствия факт doors удаляется из списка фактов задачи.

Функция modify является также весьма распространенной. Она позволяет в определенном факте поменять значение слота, например,

(deftemplate age (slot value))
(assert (age (value young)))
(modify 0 (value old))

Следующий пример описывает представление данных в виде фактов, объектов и глобальных переменных. Примеры фактов:

(voltage is 220 volt)
(meeting (subject "AI") (chief "Kuzin") (Room "3240"))

В первой строке приведен упорядоченный факт, во второй - неупорядоченный, в котором порядок слотов не важен.

CLIPS поддерживает следующие типы данных: integer, float, string, symbol, external-address, fact-address, instance-name, instance-address.

Пример integer: 594 23 +51 −17
Пример float: 594e2 23.45 +51.0 −17.5e−5

String — это строка символов, заключенная в двойные кавычки.
Пример string: "expert", "Phil Blake", "состояние -0", "quote=\"

CLIPS поддерживает следующие процедурные функции , реализующие возможности ветвления, организации циклов в программах и т.п.:

If - оператор ветвления;
While - цикл с предусловием;
loop-for-count - итеративный цикл;
prong - объединение действий в одной логической команде;
prong$ - выполнение набора действий над каждым элементом поля;
return - прерывание функции, цикла, правила и т.д.;
break - то же, что и return, но без возвращения параметров;
switch - оператор множественного ветвления;
bind - создание и связывание переменных.

Среди логических функций (возвращающих значения true или false) следует выделить следующие группы:

1) функции булевой логики: and, or, not
2) функции сравнения чисел: =, , >, , <,
3) предикативные функции для проверки принадлежности проверяемому типу: integerp, floatp, stringp, symbolp, pointerp (относится ли аргумент к xternal-address), numberp (относится ли аргумент к integer или float), lexemerp (относится ли аргумент к string или symbol), evenp (проверка

целого а четность), oddp (проверка целого на нечетность), multifildp (является ли аргумент составным полем).

4)Функции сравнения по типу и по значению: eq, neq

Среди математических функций следует выделить следующие группы:

1) Стандартные: +, -, *, /, max, min, div (целочисленное деление), abs (абсолютное значение), float (преобразование в тип float), integer (преобразование в тип integer)

2)Расширенные: sqrt (извлечение корня), round (округление числа), mod (вычисление остатка от деления)

3)Тригонометрические: sin, sinh, cos, cosh, tan, tanh, acos, acosh, acot, acoth, acsc, acsch, asec, asech, asin, asinh, atan, atanh, cot, coth, csc, csch, sec, sech, deg-grad (преобразование из градусов в секторы), deg-rad (преобразование из градусов в радианы), grad-deg (преобразование из секторов в градусы), rad-deg (преобразование из радиан в градусы)

4) Логарифмические: log, log10, exp, pi

Среди функций работы со строками следует назвать функции:

str-cat - объединение строк,

sym-cat - объединение строк в значение типа symbol,

sub-string - выделение подстроки,

str-index - поиск подстроки,

eval - выполнение строки в качестве команды CLIPS,

build - выполнение строки в качестве конструктора CLIPS,

upcase - преобразование символов в символы верхнего регистра,

lowcase - преобразование символов в символы нижнего регистра,

str-compare - сравнение строк,

str-length - определение длины строки,

check-syntax - проверка синтаксиса строки,

string-to-field - возвращение первого поля строки.

Функции работы с составными величинами являются одной из отличительных особенностей языка CLIPS. В их число входят:

create$ - создание составной величины,

nth$ - получение элемента составной величины,

members - поиск элемента составной величины,

subset$ - проверка одной величины на подмножество другой,

delete$ - удаление элемента составной величины,

explode$ - создание составной величины из строки,

implode$ - создание строки из составной величины,

subseq$ - извлечение подпоследовательности из составной величины,

replace$ - замена элемента составной величины,

insert$ - добавление новых элементов в составную величину,

first$ - получение первого элемента составной величины,

rest$ - получение остатка составной величины,

length$ - определение числа элементов составной величины,

delete-member$ - удаление элементов составной величины,

replace-member$ - замена элементов составной величины.

Функции ввода-вывода используют следующие логические имена устройств:

Stdin - устройство ввода,
stdout - устройство вывода,
wclips - устройство, используемое как справочное,
wdialog - устройство для отправки пользователю сообщений,
wdisplay - устройство для отображения правил, фактов и т.,п.,
werror - устройство вывода сообщений об ошибках,
wwarning - устройство для вывода предупреждений,
wtrase - устройство для вывода отладочной информации,

Собственно функции ввода-вывода следующие:
Open - открытие файла (виды доступа r, w, r+, a, wb),
Close - закрытие файла,
Printout - вывод информации на заданное устройство,
Read - ввод данных с заданного устройства,
Readline - ввод строки с заданного устройства,
Format - форматированный вывод на заданное устройство,
Rename - переименование файла,
Remove - удаление файла.

Среди двух десятков команд CLIPS следует назвать основные команды при работе со средой CLIPS:

Load - загрузка конструкторов из текстового файла,
load+ - загрузка конструкторов из текстового файла без отображения,
reset - сброс рабочей памяти системы CLIPS,
clear - очистка рабочей памяти системы,
run - выполнение загруженных конструкторов,
save - сохранение созданных конструкторов в текстовый файл,
exit - выход из CLIPS.

Обзор ресурсов по данной тематике. Во всемирной "паутине" Internet по данной тематике можно найти не так уж много ресурсов. Причем русскоязычные материалы по CLIPS, как и многие англоязычные, представлены только обсуждениями на специализированных форумах. К сожалению, из подобных обсуждений в большинстве случаев можно почерпнуть очень мало полезной информации. Из наиболее информативных форумов, где так или иначе затрагивалась тема CLIPS, наиболее удачным можно считать форум ресурса Библиотека программиста. Там представлены ссылки на наиболее содержательные (англоязычные) ресурсы.

Говоря об англоязычных сайтах, посвященных CLIPS, чаще всего ссылаются на ресурс http://www.ghg.net/clips/. Здесь размещена вводная информация о том, что представляет собой CLIPS как язык программирования, история его создания и развития, информация о его использовании при создании систем искусственного интеллекта и экспертных систем, приведены ответы на часто задаваемые вопросы по

CLIPS, информация о версиях и обновлениях CLIPS. Имеется также перечень серверов, связанных ресурсов, электронных конференций и подписок по CLIPS. Кроме того, отсюда можно бесплатно скачать некоторую документацию по CLIPS, программные коды и исполнимые файлы.

Еще один ресурс http://www.siliconvalleyone.com/ — сайт фирмы "Silicon Valley One". Отдельная страница посвящена информации и новостям CLIPS. Предоставлен перечень ссылок на ресурсы, где можно найти:

- архив версий CLIPS, всевозможные патчи, ссылки;
- документацию, программные коды, новейшие обновления, ссылки;
- документ, посвященный параллелизации программирования на CLIPS и подобных языках, основанных на правилах, с перечнем релевантных ссылок по исследованиям в данной области;
- документация по включению CORBA-агента в CLIPS-проекты для расширения их возможностей;
- библиография литературы по CLIPS;
- сопутствующая информация о языке, основанном на правилах — OPS-2000 и параллельных продукционных системах (PPS — Parallel Production Systems), включая документацию и примеры программ.

Еще один интересный и довольно содержательный сайт — CLIPS DLL Homepage, на котором можно найти:

- страницы, посвященные CLIPS и FuzzyCLIPS;
- документацию по CLIPS в формате PDF;
- архив CLIPS и список рассылок, а также ссылку на форум разработчиков систем на CLIPS;
- готовые проекты на CLIPS, демо-версии программ, примеры (всего более 50);
- статьи и документацию по CLIPS DLL

В завершение следует иметь в виду, что CLIPS может не удовлетворительно работать в реальном времени, когда потребуется время реакции менее 0,1 сек. В этом случае надо исследовать на разработанном прототипе механизмы вывода для всего множества правил предметной области на различных по производительности компьютерах. Как правило, современные мощные компьютеры Intel обеспечивают работу с продукционными системами объемом 1000--2000 правил в реальном времени. Веб-ориентированные средства на базе JAVA (системы Exsys Corvid, JESS) являются более медленными, чем, например, CLIPS 6.0 или OPS-2000. Поэтому CLIPS - лучший на сегодня выбор для работы в реальном времени среди распространяемых свободно оболочек ЭС, разработанных на C++.

Литература

Бабенко К. И. Основы численного анализа. М.: Наука. Физматлит. — 1986.
Бахвалов Н.С., Жидков Н.П., Кобельков Г. М. Численные методы. М.: Наука. Физматлит. — 1987.
Воднев В.Т., Наумович А.Ф., Наумович Н.Ф. Основные математические формулы. Минск: Вышэйшая школа. — 1988.
Гантмахер Ф. Теория матриц. М.: Наука. Физматлит. — 1988.
Гультяев А. Визуальное моделирование в среде MATLAB: учебный курс. СПб.: ПИТЕР. - 2001.
Джексон П. Введение в экспертные системы.: Пер. с англ.: Уч. пос. — М.: Изд. дом «Вильямс», 2001. — 624 с.
Дж. Дэннис, Р. Шнабель. Численные методы безусловной оптимизации и решения нелинейных уравнений. Пер. с англ. под ред. Ю. Г. Евтушенко. М.: Мир. — 1988.
Дьяконов В. П. Компьютерная математика. Теория и практика. М.: Нолидж. — 2001.
Дьяконов В. Как выбрать математическую систему? Монитор-Аспект, № 2, 1993.
Дьяконов В.П. Справочник по применению системы PC MatLAB. М.: Наука, Физматлит. — 1993.
Дьяконов В. П., Абраменкова И. В. MATLAB 5.0/5.3. Система символьной математики. М.: Нолидж. — 1999.
Дьяконов В. П., Абраменкова И. В., Круглов В. В. MATLAB 5 с пакетами расширений. М.: Нолидж. — 2001.
Дьяконов В. П. MATLAB. Учебный курс. СПб.: ПИТЕР. - 2001.
Дьяконов В. П. MATLAB 6. Учебный курс. СПб.: ПИТЕР. - 2001.
Дьяконов В. П. Simulink 4. Специальный справочник. СПб.: ПИТЕР. — 2002.
Дьяконов В. П., Круглов В. В. Математические пакеты расширения MATLAB. Специальный справочник. СПб.: ПИТЕР. — 2001.
Дьяконов В. П., Круглов В. В. MATLAB. Анализ, идентификация и моделирование систем. Специальный справочник. СПб.: ПИТЕР. — 2002.
Дьяконов В. П., Абраменкова И. В. MATLAB. Обработка сигналов и изображений. Специальный справочник. СПб.: ПИТЕР. — 2002.
Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5. Основы применения. М.: Солон-Пресс. — 2002.
Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5 в математике и моделировании. М.: Солон-Пресс. — 2003.
Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5. Обработка сигналов и изображений. М.: Солон-Пресс. — 2004.
Дьяконов В. П. VisSim + Mathcad + MATLAB. Визуальное математическое моделирование. М.: Солон-Пресс. — 2004.
Дж. Дэвенпорт, И. Сирэ, Э. Турнье. Компьютерная алгебра. Системы и алгоритмы алгебраических вычислений. М.: Мир. — 1991.

Г. Корн, Т. Корн. Справочник по математике для научных работников и инженеров. М.: Наука. — 1973.
Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. М.: Горячая линия-Телеком. — 2001.
Круглов В. В., Дли М. И., Голунов Р. Ю. Нечеткая логика и искусственные нейронные сети. М.: Наука. Физматлит. — 2001.
Круглов В. В., Борисов В. В. Гибридные нейронные сети. Смоленск.: Русич. — 2001.
Лазарев Ю. Ф. MatLAB 5.X. (серия «Библиотека студента») К.: Издательская группа BHV. — 2000.
Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. СПб.: БХВ-Петербург. - 2003.
Львовский Е. Н. Статистические методы построения эмпирических формул. М.: Высшая школа. — 1988.
Марчук Г.И. Методы вычислительной математики. М.: Наука. Физматлит. — 1989.
Мартынов Н.Н., Иванов А. П. MATLAB 5.x. Вычисления, визуализация, программирование. М.: КУДИЦ-ОБРАЗ. — 2000
Медведев В. С, Потемкин В. Г. Control System Toolbox. MATLAB 5 для студентов. М.: ДИАЛОГ-МИФИ. - 2004.
Потемкин В. Г. Система MATLAB. Справочное пособие. М.: ДИАЛОГ-МИФИ. - 1997.
Потемкин В. Г. MATLAB 5 для студентов. М.: ДИАЛОГ-МИФИ. - 1998.
Потемкин В. Г. Система инженерных и научных расчетов MATLAB 5.x. В 2-х т. М.: ДИАЛОГ-МИФИ. - 1999.
Потемкин В. Г. Вычисления в среде MATLAB: ДИАЛОГ-МИФИ. — 2004.
Математический энциклопедический словарь. Под редакцией Ю. В. Прохорова. М.: Советская энциклопедия. — 1988.
Рудаков П. И., Сафонов В. И. Обработка сигналов и изображений. MAT-LAB 5.x. /Под общей редакцией к.т.н. В. Г. Потемкина. М.: ДИАЛОГ-МИФИ. - 2000.
Топчеев Ю. И. Атлас для проектирования систем автоматического регулирования. М.: Машиностроение. — 1989.
Трофимов В. База данных+CLIPS=База знаний// Компьютеры+программы. — 2003. — N 10. — С. 56–61
Трофимов В.Е. Фрейм как высшая стадия структурирования// Компьютеры+программы. — 2004. — N 7. — С. 45–54
Трофимов В.Е. Автоматизация процесса диагностики РЭА на основе метода эвристической классификации// Технология и конструирование в электронной аппаратуре. — 2004. — N 2. — С. 18–24
Чен К., Джиблин П., Ирвинг А. MATLAB в математических исследованиях. М.: Мир. - 2001.
Черных И. В. SIMULINK. Среда создания инженерных приложений. М.: ДИАЛОГ-МИФИ. - 2003.

Spiegel, Murray R. Mathematical Handbook of Formulas and Tables.. New York: McGraw Hill Book Company, 1968.
MATLAB. The Language of Technical Computing. Getting Started with MATLAB. The Math Works, Inc. USA. - 2000.
MATLAB. The Language of Technical Computing. Using MATLAB. The Math Works, Inc. USA. - 2000.
MATLAB. The Language of Technical Computing. Using MATLAB Graphics. The Math Works, Inc. USA. - 2000.
MATLAB. The Language of Technical Computing. External Interfaces. The Math Works, Inc. USA. - 2000.
Simulink. Model-Based and System-Based Design. Using Simulink. The Math Works, Inc. USA. - 2002.
Simulink. Model-Based and System-Based Design. Writing S-Functions. The Math Works, Inc. USA. - 2002.
Сайт фирмы GHG Corporation, раздел CLIPS http://www.ghg.net/clips/
Сайт фирмы Silicon Valley One, раздел CLIPS
http://www.siliconvalleyone.com/clips.htm
Домашняя старница CLIPS DLL, программные разработки и документация по CLIPS http://ourworld.compuserve.com/homepages/marktoml/clipstuf.htm
http://listlib.narod.ru/robototeh.htm
http://www.kuka.com/en/
http://matlab.exponenta.ru/

В 2007 году СПбГУ ИТМО стал победителем конкурса инновационных образовательных программ вузов России на 2007–2008 годы. Реализация инновационной образовательной программы «Инновационная система подготовки специалистов нового поколения в области информационных и оптических технологий» позволит выйти на качественно новый уровень подготовки выпускников и удовлетворить возрастающий спрос на специалистов в информационной, оптической и других высокотехнологичных отраслях экономики.

КАФЕДРА ТЕХНОЛОГИИ ПРИБОРОСТРОЕНИЯ

Кафедра технологии приборостроения относиться к числу ведущих кафедр института со дня его основания в 1931 году. Тогда она называлась кафедрой механической технологии и возглавлялась известным ученым в области разработки инструмента профессором А.П. Знаменским. Позже она была переименована в кафедру технологии приборостроения.

За время своего существования кафедра выпустила из стен института более тысячи квалифицированных инженеров, более сотни кандидатов и докторов наук. В разные годы ее возглавляли известные ученые и педагоги профессора Николай Павлович Соболев и Сергей Петрович Митрофанов.

Кафедра имеет выдающиеся научные достижения. Заслуженными деятелями науки и техники РСФСР, профессором С.П. Митрофановым были разработаны научные основы группового производства, за что он был удостоен Ленинской премии СССР. Методы группового производства с успехом применяются в промышленности и постоянно развиваются его учениками. Заслуженным дея-телем науки и техники РСФСР, Заслуженным изобретателем СССР Юрием Григорьевичем Шнейдером разработаны метод и инструментарий нанесения регулярного микрорельефа на функциональной поверхности.

Основными научными направлениями кафедры являются: научные основы организации группового производства (руководители: Д.Д. Куликов, Б.С. Падун); автоматизация технологической подготовки производства (руководители: Д.Д. Куликов, Б.С. Падун, Е.И. Яблочников); регуляризация микрорельефа поверхностей деталей машин и приборов(руководитель Ю.П. Кузьмин); управление функциональными свойствами поверхностного слоя деталей (руководитель В.А. Валетов). Последнее направление организованно в 1988 году и в настоящее время детально разработаны не только теоретические основы управления характеристиками поверхностного слоя деталей, но и методики проведения экспериментальных исследований по

определению влияния различных факторов на характеристики поверхностного слоя.

Разработаны современные установки – измерительно-вычислительные ком-плексы для анализа микрогеометрии поверхностей и технологических остаточ-ных напряжений в поверхностном слое. По этому направлению за последние годы защищено три кандидатских диссертации и подготовлена одна доктор-ская. В настоящее время на кафедре в рамках этого направления работают не только преподаватели, но и молодые аспиранты.

Кафедра имеет тесные научные и учебные связи с университетами Германии, Франции, Китая. Наиболее способные студенты и аспиранты проходят стажировку по интересующим их проблемам в Техническом университете г. Ильменау (Германия).

Кафедра технологии приборостроения была создана как ведущая и обслуживала чтением лекций по курсу “Технология приборостроения” все выпускающие кафедры ВУЗа. На заведование кафедрой был приглашен профессор Знаменский А.П. – главный инженер завода ГОМЗ, автор первого “Справочника Металлиста”, известного во многих странах мира. Преподавателями и сотрудниками кафедры были приглашены высококвалифицированные специалисты, в основном работники промышленности – проф. Соколов - главный инженер инструментального завода им. Васкова, Буталов В.И., Бельфир, Казак и др. Нужно отметить, что до 1961года кафедра была слабо оснащена оборудованием и поэтому большое время уделялось прохождению практики непосредственно на заводах “ГОМЗ”, “Красногвардеец”, “Пишмаш” и др., где давались хорошая подготовка, как по работе на станках, так и по разработке грамотных технологических процессов. В 1951г. в кафедру влились кафедры станков и теории резания. После смерти А.П.Знаменского заведующими кафедрой были сотрудники института проф. Барун В.А., Маталин А.А., Соболев Н.П., научные труды которых использовались в качестве учебников. Учитывая длительный срок эксплуатации, оборудование устарело и было изношено. При кафедре имелись небольшие мастерские (25 токарных станков “ДИП”, 3 фрезерных станка “Красный пролетарий” и другое вспомогательное оборудование). По своей оснащенности она не удовлетворяла качества подготовки инженеров-технологов. На кафедре ученое звание профессора имел только Н.П.Соболев – заведующий кафедрой.

В 1962 году, после смерти профессора Соболева Н.П. заведующим кафедрой был избран лауреат Ленинской премии, д.т.н., профессор С.П.Митрофанов, прошедший школу по всем этапам производства: механика, начальника технологического и конструкторского бюро, главного технолога завода, начальника производства и главного инженера завода “ГОМЗ”. В течение шести лет, с 1954 по 1961 год, он курировал всю промышленность и науку города и Ленинградской области, работая секретарем Ленинградского ОК КПСС.

Кроме учебных лабораторий в 1964 году была создана отраслевая лаборатория по технологии и организации группового производства, ведущая НИР со многими предприятиями не только страны, но и ГДР, ЧССР, Болгарией, Венгрией, Великобританией и др. Кафедра и ее лаборатории стали поставщиком кадров не только для нашего ВУЗа, но и ВУЗов других регионов. За время работы с 1964 года подготовлено более 100 к.т.н. и несколько д.т.н.

В настоящее время на кафедре работают 7 профессоров и 9 доцентов.

С.П.Митрофанов оставил заведование кафедрой в связи с возрастом, оставаясь профессором кафедры. С 1998 года кафедру возглавлял проф. Н.Д.Фролов. В настоящий момент кафедру возглавляет Е.И. Яблочников.

Вячеслав Алексеевич Валетов
Юрий Петрович Кузьмин
Анна Алексеевна Орлова
Сергей Дмитриевич Третьяков

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ СРС ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИЯ ПРИБОРОСТРОЕНИЯ»

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел
Санкт-Петербургского государственного университета информационных технологий механики и оптики
Лицензия ИД №00408 от 05.11.99

Зав.РИО Н.Ф. Гусарова

Объём 32 стр. Тираж 100 экз. Подписано к. печати“27.11.2008г
Закакз № 1464

Редакционно-издательский отдел
Санкт-Петербургского государственного
университета информационных
технологий, механики и оптики
197101, Санкт-Петербург, Кронверкский пр., 49