Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ И ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

сборник учебно-методических материалов

для направления подготовки 38.03.05 «Бизнес-информатика»

Благовещенск, 2017

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Составитель: Акилова И.М.

Функциональное программирование и интеллектуальные системы: сборник учебно-методических материалов для направления подготовки для направления подготовки 38.03.05 «Бизнес информатика». – Благовещенск: Амурский гос. ун-т, 2017.

КРАТКОЕ ИЗЛОЖЕНИЕ ЛЕКЦИОННОГО МАТЕРИАЛА

Лекция № 1 **Тема:** ***История развития искусственного интеллекта.***

Под искусственным интеллектом понимают научное направление, в рамках которого ставятся и решаются задачи аппаратного и программного моделирования тех видов человеческой деятельности, которые традиционно считаются интеллектуальными.

Свойство человека решать любые творческие задачи, обучаться, объяснять решение, представлять в памяти знания об окружающем мире выделяет его из всего живого мира. Являясь универсальной биомашиной по обработке информации он все-таки сталкивается с определенными трудностями по восприятию и анализу больших потоков информации, особенно в последнее время.

Этапы развития искусственного интеллекта

1. Кибернетики того времени пытались построить машины, моделирующие человеческий мозг. Появление первых электронных вычислительных машин, способных выполнять вычисления больших чисел десятки тысяч раз в секунду придавало очень большой оптимизм в исследованиях по искусственному интеллекту.

Розенблатом был создан самоорганизуюшийся автомат - PERCEPTRON, который считался грубой моделью сетчатки глаза. Его можно было научить распознавать образы, хотя их количество ограничивалось. Энтузиазм в отношении разработок подобных систем основывался на кибернетических представлениях Норберта Винера и Уоррена Маккалока об абстрактных нейронных сетях. Считалось, что можно взять сильно связанную систему модельных нейронов, которой раньше ничего не было известно и обучить ее тому, что задумал создатель методом поощрений и наказаний. При этом тот факт, что мозг человека содержит 10 в 10 степени нейронов тактично обходился.

Причиной появления представлений об абстрактных нейронных сетях являлись достижения в психологии и нейрофизиологии того времени. Ученые уже имели представления о строении нейронов и их взаимосвязи. Развитие электроники привело к попыткам реализовать схемное представление нейрона и нейронных сетей.

Но особого развития этого направления в искусственном интеллекте в те времена не произошло, все таки и техника была еще примитивна, и исследования переключились на исследование других методов имитации разумного поведения при решении задач. К тому же Минский и Пэйперт провели теоретическую работу и показали что возможности перцепторна ограниченны, а экспериментальные результаты оказались недостаточно хорошими.

Исследования нейронных сетей практически прекратилось, но не совсем. В настоящее время теория нейронных сетей получила новое развитие.

2. На новые рубежи в исследовании искусственного интеллекта указали Аллен Ньюэлл и Герберт Саймон из университета Карнеги-Меллона (США). Центральным для их подхода явилось представление об эвристическом поиске. Они считали что мышление человека основано на сочетании простых задач манипулирования символами - сравнение, поиск, модификация и т.п., а следовательно эти задачи может выполнить и компьютер.

Они достигли реальных практических результатов т.к. уровень развития компьютеров того времени позволял это сделать. В конце 50-х годов появились первые эвристические программы: ЛОГИК - ТЕОРЕТИК, предназначенный для доказательства теорем в исчислении высказываний и ОБЩИЙ РЕШАТЕЛЬ ЗАДАЧ. То что их программы действительно моделируют человеческое мышление они доказывали, сравнивая рассуждения при доказательстве теоремы думающего вслух человека. В начале 70-х они даже предложили общую методику составления программ, моделирующих мышление и создали систему GPS. Эта система была универсальной в том отношении, что "не было конкретного указания, к какой предметной области относится данная задача". Но эта универсальность относилась только к ограниченной области математических головоломок: Ханойская башня, проблема миссионеров и туземцев и др., которые с точки зрения людей проблемами не являются.

Основой системы являлся "поиск в глубину" и процесс постепенного разбиения задачи на подзадачи, пока не будет получена легко решаемая подзадача.

3. Большое значение на дальнейшее развитие ИИ оказало появление метода резолюций предложенный Робинсоном, который является исчерпывающим методом доказательства в логике предикатов. Именно в это время появляется язык программирования нового типа - PROLOG, (программирование логики), который является одним из универсальных языков программирования задач искусственного интеллекта.

Постепенно интерес к поиску эффективных и универсальных эвристик переместился в сторону разработок интеллектуальных задач с узкой предметной областью. В 70-х годах начали появляться первые экспертные системы.

Экспертные системы стали первым реальным практическим применением искусственного интеллекта при решении интеллектуальных задач, решение которых с позиций алгоритмического программирования не имеет смысла. Они ознаменовали собой третий этап развития искусственного интеллекта.

Первые системы: MYSIN и DENDRAL - давали результаты сравнимые с экспертами среднего класса. MYSIN была родоначальником целой серии медико-диагностических машин, используемых в клинической практике. Предназначенная для диагностики инфекционных заболеваний, она приводила правдоподобные заключения даже если не все исходные данные были верны. Имитируя рассуждения эксперта, она оперировала с "неточными" знаниями и могла объяснить процесс своих "рассуждений". Она выполняла работу, которой люди обучаются годами, и хотя обладала ограниченными возможностями по сравнению даже с плохим врачом по объему разносторонних знаний, работа ее оставляла большое впечатление. По сути дела она стала эталоном для создания большинства экспертных систем.

Созданная позже система PROSPECTOR, работающая в области геологии, открыла ранее неизвестные запасы молибдена. Система PUFF – созданная по подобию MYSIN и диагностирующая дыхательные заболевания, до сих пор используется в Тихоокеанском центре вблизи Сан-Франциско.

Основное внимание в ИИ переместилось с моделирования человеческого мышления на разработку машинно-ориентированных методов решения задач.

Целью ИИ стала разработка программ, способных решать "человеческие задачи", с которыми до недавнего времени мог справиться только человек.Экспертные системы явились причиной необузданного оптимизма в практических разработках интеллектуальных программ 70-х годов. Но основная трудность с которой начали сталкиваться при разработке экспертных систем - это получение знаний. Именно обширность и качество базы знаний и определяют успех экспертной системы. Получение знаний от экспертов оказалось очень не легкой задачей. Появилась даже отдельная профессия - инженер по знаниям (аналитик знаний) который формализует суждения человека-эксперта в один из методов представления знаний искусственного интеллекта.

Новое направление в развитии искусственного интеллекта стали связывать с методами получения знаний самой интеллектуальной системой. Дуг Ленат из Стэндфордского университета создал машинную обучающуюся систему EURISKO, которая автоматически улучшает и расширяет свой запас эвристических правил. Эта система выигрывала три года подряд в учебной военной игре (несмотря на то что правила игры каждый раз менялись, чтобы помешать ей это сделать), она произвела переворот в области создания СБИС (сверхбольших интегральных схем), изобретя трехмерный узел типа И/ИЛИ. Появление программ автоматической индукции подобным EURISKO, является важнейшим этапом в развитии искусственного интеллекта в последние десять лет. До сих пор перенесение умения специалиста-человека в машинную программу есть утомительная и долгая процедура. Таким образом искусственный интеллект 80-х перенес внимание в область проблем машинного обучения.

Искусственный интеллект, двигаясь в своем развитии по спирали замкнул круг и

поставил перед исследователями задачу, которую пытались решить первые исследователи искусственного интеллекта - машинное обучение и накопление знаний, решение которых в настоящее время опирается на более мощную технику и на более широкие потребности общества нежели это было раньше.

Говоря об истории искусственного интеллекта, нельзя не вспомнить и то, какой вклад внесла Япония в его развитие. Благодаря ей исследования в этой области получили большую поддержку во многих ведущих странах мира. Речь идет об японском проекте компьютеров 5-го поколения, о котором Япония объявила в 1978 году.

Министерство внешней торговли и промышленности Японии поручило электротехнической лаборатории выработать проект вычислительных систем 90-х годов. Предполагалось, что проект не должен конкурировать с направлением, выбранным фирмой IBM. Формально к проекту приступили в 1982 году. Проект машин 5-го поколения охватывает весьма широкую область. Его задачей является создание и программного обеспечения и аппаратная реализация архитектуры.

В проекте было выделено 7 направлений исследований:

Вычислительная система 5-го будет ориентированна на обработку знаний, обладать возможностями логического вывода и развитым интеллектуальным человеко-машинным интерфейсом.

На первом этапе создается персональная последовательная машина вывода. В дальнейшем появляется высокопроизводительная параллельная машина. Быстродействие будет измеряться числом логических выводов в секунду.

Намечаемая производительность: 1 ГЛВС
1 ЛВС = 100 - 100000 миллионов машинных операций в секунду.

Был создан план работ: начальный промежуточный и заключительный.
1. Создание высокопроизводительной персональной ПРОЛОГ - машины
2. Создание машины реляционной базы данных
3. Проведение фундаментальных исследований.

В 1985 году должны появится первые результаты.

В настоящее время проект не реализован, хотя его завершение должно быть в начале 90-х годах. Но его главное значение в том, что интерес к искусственному интеллекту возрос необычайно сильно. Почти во всех ведущих странах мира были созданы исследовательские группы и выделены большие средства для научных исследований.

Наиболее очевидный факт, который проявился благодаря игре в машины
пятого поколения, состоит в том, что для игры не хватает людей. В этой области до сих пор ощущается острый недостаток работников и на них очень большой спрос.

Поэтому в настоящее время возникают задачи подготовки следующего поколения исследователей, разработка специального курса и привлечений финансирующих результатов.

В настоящее время исследования искусственного интеллекта ведутся в двух направлениях: программно - прогматическое и бионическое.

Первое занимается созданием программ на компьютерах Фон-Нейманского типа, с помощью которых можно решать те задачи, решение которых до этого считалось исключительно прерогативой человека.

Программно – прогматическое направление
1. Информация о мышлении и языке:
2. Интеллектуальные программы:
3. Распознающие и узнающие программы
4. Прочие программы
5. Модели поведения
6. Программы доказательства теорем
7. Эвристические программы
8. Работа со знаниями:
9. Интеллектуальное программирование

10. Автоматический синтез программ
11. Инструментальные системы
12. Интеллектуальные программные системы
13. Интеллектуальные информационные системы
14. Интеллектуальные системы проектирования и научных исследований
15. Обучающие системы

Бионическое направление

- информация о морфологической (нейрофизиологической) структуре;
- нейробионической подход;
- информация о целостных структурах организма;
- структурно-эвристический;
- информация о функциональных механизмах организма;
- гомеостатический.

Интересуется проблемами искусственного воспроизведения тех структур и процессов, которые характерны для живого человеческого мозга и которые лежат в основе процесса решения задач человеком. Это направление имеет четко выраженный фундаментальный характер, и его интенсивное развитие невозможно без одновременного глубокого изучения мозга нейрофизиологическими, морфологическими и психологическими методами.

Буквально за последние несколько лет интерес к ИИ возрос по ряду объективных причин. Самой важной из них является информационная революция.

Развитие средств вычислительной техники за последнее пятилетие создало предпосылки для реализации многих исследовательских разработок, которые не имели большого практического применения из-за недостаточно развитой вычислительной техники.

В ведущих странах мира произошел значительный перенос информации в память компьютеров и ее обработка. Общество уже требует от ЭВМ не только данных для решения задач, но и того как их решать.

Поэтому применение методов ИИ, наделение программ интеллектуальными функциями поведения является закономерным этапом развития информационного общества. Решение проблемы создает новые проблемы и так далее.

В настоящее время областью массового применения средств ИИ стали системы:

- Обработки ЕЯ грамматик
- Распознавание речи и изображения
- Системы машинного перевода
- Экспертные системы.

Последние являются целым отдельным направлением в рамках исследований по ИИ. Именно в них получено очень большое число практических разработок.

Исследования в ЭС еще называют инженерией знаний. В задачу этого направления входят исследование и разработка программ, использующих знания и процедуры вывода для решения задач, являющихся трудными для людей - экспертов.

Можно дать определение ЭС: это программные комплекса, накапливающие опыт специалистов в некоторой предметной области в целях тиражирования для консультации менее подготовленных пользователей.

Огромный интерес к ЭС вызван тремя причинами:

1. Они ориентированны на решение широкого круга слабо формализованных задач, которые считались недоступными для компьютеров.
2. Специалисты, не знающие программирования могут самостоятельно разрабатывать ЭС при помощи оболочек ЭС.
3. ЭС при решении практических задач достигают результатов, не уступающих, а иногда и превосходящих возможности людей-экспертов.

Лекции № 2,3 **Тема: Представление знаний в интеллектуальных системах.**

В настоящее время в исследованиях по искусственному интеллекту (ИИ) выделились шесть направлений:

1. Представление знаний.
2. Манипулирование знаниями.
3. Общение.
4. Восприятие.
5. Обучение.
6. Поведение.

В рамках направления "Представление знаний" решаются задачи, связанные с формализацией и представлением знаний в памяти интеллектуальной системы (ИС). Для этого разрабатываются специальные модели представления знаний и языки для описания знаний, выделяются различные типы знаний. Изучаются источники, из которых ИС может черпать знания, и создаются процедуры и приемы, с помощью которых возможно приобретение знаний для ИС. Проблема представления знаний для ИС чрезвычайно актуальна, т.к. ИС - это система, функционирование которой опирается на знания о проблемной области, которые хранятся в ее памяти.

Данные и знания. Основные определения.

Информация, с которой имеют дело ЭВМ, разделяется на *процедурную* и *декларативную*. Процедурная информация овеществлена в *программах*, которые выполняются в процессе решения задач, декларативная информация - в *данных*, с которыми эти программы работают. Стандартной формой представления информации в ЭВМ является *машинное слово*, состоящее из определенного для данного типа ЭВМ числа двоичных разрядов - *битов*. Машинное слово для представления данных и машинное слово для представления команд, образующих программу, могут иметь одинаковое или разное число разрядов. В последнее время для представления данных и команд используются одинаковые по числу разрядов машинные слова.

Содержимое памяти образует *информационную базу*. По мере развития исследований в области ИС возникла концепция знаний, которые объединили в себе многие черты процедурной и декларативной информации. *База знаний* - необходимая составляющая программного комплекса ИИ. Машины, реализующие алгоритмы ИИ, называются *машинами, основанными на знаниях*, а подраздел теории ИИ, связанный с построением экспертных систем, - *инженерией знаний.*

Особенности знаний:

1. *Внутренняя интерпретируемость*. Каждая информационная единица должна иметь уникальное имя, по которому ИС находит ее, а также отвечает на запросы, в которых это имя упомянуто. Когда данные, хранящиеся в памяти, были лишены имен, то отсутствовала возможность их идентификации системой. Данные могла идентифицировать лишь программа, извлекающая их из памяти по указанию программиста, написавшего программу. Что скрывается за тем или иным двоичным кодом машинного слова, системе было неизвестно.

Таблица 1

Фамилия	Год рождения	Специальность	Стаж, число лет
Попов	1965	Слесарь	5
Сидоров	1946	Токарь	20
Иванов	1925	Токарь	30
Петров	1937	Сантехник	25

Если, например, в память ЭВМ нужно было записать сведения о сотрудниках учреждения, представленные в табл. 1, то без внутренней интерпретации в память ЭВМ была бы занесена совокупность из четырех машинных слов, соответствующих строкам этой таблицы. При этом информация о том, какими группами двоичных разрядов в этих машинных словах закодированы сведения о специалистах, у системы отсутствуют. Они известны лишь программисту, который использует данные табл. 1 для решения

возникающих у него задач. Система не в состоянии ответить на вопросы типа "Что тебе известно о Петрове?" или "Есть ли среди специалистов сантехник?".

При переходе к знаниям в память ЭВМ вводится информация о некоторой *протоструктуре информационных единиц*. В рассматриваемом примере она представляет собой специальное машинное слово, в котором указано, в каких разрядах хранятся сведения о фамилиях, годах рождения, специальностях и стажах. При этом должны быть заданы специальные словари, в которых перечислены имеющиеся в памяти системы фамилии, года рождения, специальности и продолжительности стажа. Все эти *атрибуты* могут играть роль имен для тех машинных слов, которые соответствуют строкам таблицы. По ним можно осуществлять поиск нужной информации. Каждая строка таблицы будет экземпляром протоструктуры. В настоящее время СУБД обеспечивают реализацию внутренней интерпретируемости всех информационных единиц, хранящихся в базе данных.

2. Структурированность. Информационные единицы должны обладать гибкой структурой. Для них должен выполняться "принцип матрешки", т.е. рекурсивная вложимость одних информационных единиц в другие. Каждая информационная единица может быть включена в состав любой другой, и из каждой информационной единицы можно выделить некоторые составляющие ее информационные единицы. Другими словами, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа "часть - целое", "род - вид" или "элемент - класс".

3. *Связность*. В информационной базе между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Прежде всего эти связи могут характеризовать отношения между информационными единицами. Семантика отношений может носить декларативный или процедурный характер. Например, две или более информационные единицы могут быть связаны отношением "одновременно", две информационные единицы - отношением "причина - следствие" или отношением "быть рядом". Приведенные отношения характеризуют декларативные знания. Если между двумя информационными единицами установлено отношение "аргумент - функция", то оно характеризует процедурное знание, связанное с вычислением определенных функций. Далее будем различать *отношения структуризации, функциональные отношения, каузальные отношения* и *семантические отношения*. С помощью первых задаются иерархии информационных единиц, вторые несут процедурную информацию, позволяющую находить (вычислять) одни информационные единицы через другие, третьи задают причинно - следственные связи, четвертые соответствуют всем остальным отношениям.

Между информационными единицами могут устанавливаться и иные связи, например, определяющие порядок выбора информационных единиц из памяти или указывающие на то, что две информационные единицы несовместимы друг с другом в одном описании.

Перечисленные три особенности знаний позволяют ввести общую модель представления знаний, которую можно назвать *семантической сетью*, представляющей собой иерархическую сеть, в вершинах которой находятся информационные единицы. Эти единицы снабжены индивидуальными именами. Дуги семантической сети соответствуют различным связям между информационными единицами. При этом иерархические связи определяются отношениями структуризации, а неиерархические связи - отношениями иных типов.

4. Семантическая метрика. На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее ситуационную близость информационных единиц, т.е. силу ассоциативной связи между информационными единицами. Его можно было бы назвать *отношением релевантности* для информационных единиц. Такое отношение дает возможность выделять в информационной базе некоторые типовые ситуации (например, "покупка", "регулирование движения на перекрестке").

Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

5. *Активность*. С момента появления ЭВМ и разделения используемых в ней информационных единиц на данные и команды создалась ситуация, при которой данные пассивны, а команды активны. Все процессы, протекающие в ЭВМ, инициируются командами, а данные используются этими командами лишь в случае необходимости. Для ИС эта ситуация не приемлема. Как и у человека, в ИС актуализации тех или иных действий способствуют знания, имеющиеся в системе. Таким образом, выполнение программ в ИС должно инициироваться текущим состоянием информационной базы. Появление в базе фактов или описаний событий, установление связей может стать источником активности системы.

Перечисленные пять особенностей информационных единиц определяют ту грань, за которой данные превращаются в знания, а базы данных перерастают в *базы знаний* (БЗ). Совокупность средств, обеспечивающих работу с знаниями, образует *систему управления базой знаний* (СУБЗ). В настоящее время не существует баз знаний, в которых в полной мере были бы реализованы внутренняя интерпретируемость, структуризация, связность, введена семантическая мера и обеспечена активность знаний.

Модели представления знаний. Неформальные (семантические) модели.

Существуют два типа методов представления знаний (ПЗ):

1. Формальные модели ПЗ;
2. Неформальные (семантические, реляционные) модели ПЗ.

Очевидно, все методы представления знаний, которые рассмотрены выше, включая продукции (это система правил, на которых основана продукционная модель представления знаний), относятся к неформальным моделям. В отличие от формальных моделей, в основе которых лежит строгая математическая теория, неформальные модели такой теории не придерживаются. Каждая неформальная модель годится только для конкретной предметной области и поэтому не обладает универсальностью, которая присуща моделям формальным. Логический вывод - основная операция в СИИ - в формальных системах строг и корректен, поскольку подчинен жестким аксиоматическим правилам. Вывод в неформальных системах во многом определяется самим исследователем, который и отвечает за его корректность.

Каждому из методов ПЗ соответствует свой способ описания знаний.

1. *Логические модели.* В основе моделей такого типа лежит *формальная система*, задаваемая четверкой вида: $M = <T, P, A, B>$. Множество T есть *множество базовых элементов* различной природы, например слов из некоторого ограниченного словаря, деталей детского конструктора, входящих в состав некоторого набора и т.п. Важно, что для множества T существует некоторый способ определения принадлежности или не принадлежности произвольного элемента к этому множеству. Процедура такой проверки может быть любой, но за конечное число шагов она должна давать положительный или отрицательный ответ на вопрос, является ли x элементом множества T. Обозначим эту процедуру П(T). Множество P есть множество *синтаксических правил*. С их помощью из элементов T образуют *синтаксически правильные совокупности*. Например, из слов ограниченного словаря строятся синтаксически правильные фразы, из деталей детского конструктора с помощью гаек и болтов собираются новые конструкции. Декларируется существование процедуры П(P), с помощью которой за конечное число шагов можно получить ответ на вопрос, является ли совокупность X синтаксически правильной. В множестве синтаксически правильных совокупностей выделяется некоторое подмножество A. Элементы A называются *аксиомами*. Как и для других составляющих формальной системы, должна существовать процедура П(A), с помощью которой для любой синтаксически правильной совокупности можно получить ответ на вопрос о принадлежности ее к множеству A. Множество B есть множество *правил вывода*. Применяя их к элементам A, можно получать новые синтаксически правильные совокупности, к которым снова можно применять правила из B. Так формируется *множество выводимых* в данной формальной

системе *совокупностей*. Если имеется процедура П(*B*), с помощью которой можно определить для любой синтаксически правильной совокупности, является ли она выводимой, то соответствующая формальная система называется *разрешимой*. Это показывает, что именно правило вывода является наиболее сложной составляющей формальной системы. Для знаний, входящих в базу знаний, можно считать, что множество *A* образуют все информационные единицы, которые введены в базу знаний извне, а с помощью правил вывода из них выводятся новые *производные знания*. Другими словами формальная система представляет собой генератор порождения новых знаний, образующих множество *выводимых* в данной системе *знаний*. Это свойство логических моделей делает их притягательными для использования в базах знаний. Оно позволяет хранить в базе лишь те знания, которые образуют множество *A*, а все остальные знания получать из них по правилам вывода.

2. *Сетевые модели*. В основе моделей этого типа лежит конструкция, названная ранее семантической сетью. Сетевые модели формально можно задать в виде $H = <I, C_1, C_2, ..., C_n, \Gamma>$. Здесь *I* есть множество информационных единиц; $C_1, C_2, ..., C_n$ - множество типов связей между информационными единицами. Отображение Γ задает между информационными единицами, входящими в *I*, связи из заданного набора типов связей. В зависимости от типов связей, используемых в модели, различают *классифицирующие сети, функциональные сети* и *сценарии*. В классифицирующих сетях используются отношения структуризации. Такие сети позволяют в базах знаний вводить разные иерархические отношения между информационными единицами. Функциональные сети характеризуются наличием функциональных отношений. Их часто называют *вычислительными моделями*, т.к. они позволяют описывать процедуры "вычислений" одних информационных единиц через другие. В сценариях используются каузальные отношения, а также отношения типов "средство - результат", "орудие - действие" и т.п. Если в сетевой модели допускаются связи различного типа, то ее обычно называют семантической сетью.

3. *Продукционные модели*. В моделях этого типа используются некоторые элементы логических и сетевых моделей. Из логических моделей заимствована идея правил вывода, которые здесь называются *продукциями*, а из сетевых моделей - описание знаний в виде семантической сети. В результате применения правил вывода к фрагментам сетевого описания происходит трансформация семантической сети за счет смены ее фрагментов, наращивания сети и исключения из нее ненужных фрагментов. Таким образом, в продукционных моделях процедурная информация явно выделена и описывается иными средствами, чем декларативная информация. Вместо логического вывода, характерного для логических моделей, в продукционных моделях появляется *вывод на знаниях*.

4. *Фреймовые модели*. В отличие от моделей других типов во фреймовых моделях фиксируется жесткая структура информационных единиц, которая называется *протофреймом*. В общем виде она выглядит следующим образом:

(Имя фрейма:
Имя слота 1(значение слота 1)
Имя слота 2(значение слота 2)
. . . . . .
Имя слота К (значение слота К)).

Значением *слота* может быть практически что угодно (числа или математические соотношения, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов). В качестве значения слота может выступать набор слотов более низкого уровня, что позволяет во фреймовых представлениях реализовать "принцип матрешки".

При конкретизации фрейма ему и слотам присваиваются конкретные имена и происходит заполнение слотов. Таким образом, из протофреймов получаются *фреймы - экземпляры.* Переход от исходного протофрейма к фрейму - экземпляру может быть многошаговым, за счет постепенного уточнения значений слотов. Связи между фреймами

задаются значениями специального слота с именем "Связь". Часть специалистов по ИС считает, что нет необходимости специально выделять фреймовые модели в представлении знаний, т.к. в них объединены все основные особенности моделей остальных типов.

Формальные модели представления знаний.

Система ИИ в определенном смысле моделирует интеллектуальную деятельность человека и, в частности, - логику его рассуждений. В грубо упрощенной форме наши логические построения при этом сводятся к следующей схеме: из одной или нескольких посылок (которые считаются истинными) следует сделать "логически верное" заключение (вывод, следствие). Очевидно, для этого необходимо, чтобы и посылки, и заключение были представлены на понятном языке, адекватно отражающем предметную область, в которой проводится вывод. В обычной жизни это наш естественный язык общения, в математике, например, это язык определенных формул и т.п. Наличие же языка предполагает, во - первых, наличие алфавита (словаря), отображающего в символьной форме весь набор базовых понятий (элементов), с которыми придется иметь дело и, во - вторых, набор синтаксических правил, на основе которых, пользуясь алфавитом, можно построить определенные выражения.

Логические выражения, построенные в данном языке, могут быть истинными или ложными. Некоторые из этих выражений, являющиеся всегда истинными. Объявляются *аксиомами* (или *постулатами*). Они составляют ту базовую систему посылок, исходя из которой и пользуясь определенными правилами вывода, можно получить заключения в виде новых выражений, также являющихся истинными.

Если перечисленные условия выполняются, то говорят, что система удовлетворяет требованиям *формальной теории.* Ее так и называют *формальной системой* (ФС). Система, построенная на основе формальной теории, называется также *аксиоматической системой.* Формальная теория должна, таким образом, удовлетворять следующему определению:

всякая формальная теория $F = (A, V, W, R)$, определяющая некоторую аксиоматическую систему, характеризуется:

наличием алфавита (словаря), A,

множеством синтаксических правил, V,

множеством аксиом, лежащих в основе теории, W,

множеством правил вывода, R.

Исчисление высказываний (ИВ) и исчисление предикатов (ИП) являются классическими примерами аксиоматических систем. Эти ФС хорошо исследованы и имеют прекрасно разработанные модели логического вывода - главной метапроцедуры в интеллектуальных системах. Поэтому все, что может и гарантирует каждая из этих систем, гарантируется и для прикладных ФС как моделей конкретных предметных областей. В частности, это гарантии непротиворечивости вывода, алгоритмической разрешимости (для исчисления высказываний) и полуразрешимости (для исчислений предикатов первого порядка).

ФС имеют и недостатки, которые заставляют искать иные формы представления. Главный недостаток - это "закрытость" ФС, их негибкость. Модификация и расширение здесь всегда связаны с перестройкой всей ФС, что для практических систем сложно и трудоемко. В них очень сложно учитывать происходящие изменения. Поэтому ФС как модели представления знаний используются в тех предметных областях, которые хорошо локализуются и мало зависят от внешних факторов.

1. *Продукционные системы*

Продукции наряду с фреймами являются наиболее популярными средствами представления знаний в ИИ. Продукции, с одной стороны, близки к логическим моделям, что позволяет организовывать на них эффективные процедуры вывода, а с другой стороны, более наглядно отражают знания, чем классические логические модели. В них отсутствуют жесткие ограничения, характерные для логических исчислений, что дает возможность изменять интерпретацию элементов продукции.

Компоненты продукционных систем

В общем виде под продукцией понимается выражение следующего вида: (i); Q;P;A□B;N. Здесь i - имя продукции, с помощью которого данная продукция выделяется из всего множества продукций. В качестве имени может выступать некоторая лексема, отражающая суть данной продукции (например, "покупка книги "), или порядковый номер продукций в их множестве, хранящимся в памяти системы.

Элемент Q характеризует сферу применения продукции. Такие сферы легко выделяются в когнитивных структурах человека. Наши знания как бы "разложены по полочкам". На одной полочке хранятся знания о том, как надо готовить пищу, на другой как добраться до работы, и т.п. Разделение знаний на отдельные сферы позволяет экономить время на поиск нужных знаний. Такое же разделение на сферы в базе знаний ИИ целесообразно и при использовании для представления знаний продукционных моделей.

Основным элементом продукции является ее ядро: A□ B. Интерпретация ядра продукции может быть различной и зависит от того, что стоит слева и справа от знака секвенции □. Обычное прочтение ядра продукции выглядит так: ЕСЛИ A, ТО B, более сложные конструкции ядра допускают в правой части альтернативный выбор, например, ЕСЛИ A, ТО B1, ИНАЧЕ B2. Секвенция может истолковываться в обычном логическом смысле как знак логического следования B из истинного A (если A не является истинным выражением, то о B ничего сказать нельзя). Возможны и другие интерпретации ядра продукции, например A описывает некоторое условие, необходимое для того, чтобы можно было совершить действие B.

Элемент P есть условие применимости ядра продукции. Обычно P представляет собой логическое выражение (как правило предикат). Когда P принимает значение "истина", ядро продукции активизируется. Если P "ложно", то ядро продукции не может быть использовано.

Элемент N описывает постусловия продукции. Они актуализируются только в том случае, если ядро продукции реализовалось. Постусловия продукции описывают действия и процедуры, которые необходимо выполнить после реализации B. Выполнение N может происходить сразу после реализации ядра продукции.

Если в памяти системы хранится некоторый набор продукций, то они образуют систему продукций. В системе продукций должны быть заданы специальные процедуры управления продукциями, с помощью которых происходит актуализация продукций и выбор для выполнения той или иной продукции из числа актуализированных. В ряде систем ИИ используют комбинации сетевых и продукционных моделей представления знаний. В таких моделях декларативные знания описываются в сетевом компоненте модели, а процедурные знания - в продукционном. В этом случае говорят о работе продукционной системы над семантической сетью.

Классификация ядер продукции.

Ядра продукции можно классифицировать по различным основаниям. Прежде всего все ядра делятся на два больших типа: детерминированные и недетерминированные. В детерминированных ядрах при актуализации ядра и при выполнимости A правая часть ядра выполняется обязательно; в недетерминированных ядрах B может выполняться и не выполняться. Таким образом, секвенция □ в детерминированных ядрах реализуется с необходимостью, а в недетерминированных - с возможностью. Интерпретация ядра в этом случае может, например, выглядеть так: ЕСЛИ A, ТО ВОЗМОЖНО B.

Возможность может определяться некоторыми оценками реализации ядра. Например, если задана вероятность выполнения B при актуализации A, то продукция может быть такой: ЕСЛИ A, ТО С ВЕРОЯТНОСТЬЮ P РЕАЛИЗОВАТЬ B. Оценка реализации ядра может быть лингвистической, связанной с понятием терм - множества лингвистической переменной, например: ЕСЛИ A, ТО С БОЛЬШЕЙ ДОЛЕЙ УВЕРЕННОСТИ B. Возможны иные способы реализации ядра.

Детерминированные продукции могут быть однозначными и альтернативными. Во втором случае в правой части ядра указываются альтернативные возможности выбора, которые оцениваются специальными весами выбора. В качестве таких весов могут использоваться вероятностные оценки, лингвистические оценки, экспертные оценки и т.п.

Особым типом являются прогнозирующие продукции, в которых описываются последствия, ожидаемые при актуализации А, например: ЕСЛИ А, ТО С ВЕРОЯТНОСТЬЮ Р МОЖНО ОЖИДАТЬ В.

Лекция № 4 Тема: Обобщенная схема ЕЯ-систем. Методы реализации ЕЯ-систем. Основные классы ЕЯ-систем. Системы общения с базами данных. Обзор промышленных ЕЯ-систем.

В конце 60-х годов в исследованиях по ИИ сформировалось самостоятельное направление, получившее название "обработка естественного языка".

Задачей данного направления является исследование методов и разработка систем, обеспечивающих реализацию процесса общения с ЭВМ на естественном языке.

Сложность создания средств общения, предназначенных для конечных пользователей, обусловлена в значительной степени отсутствием единой теории языкового общения, охватывающей все аспекты взаимодействия коммуникантов.

Низкая эффективность, а часто неприемственность традиционных средств общения в большинстве случаев вызвано тем, что в них не учитываются важнейшие особенности процесса общения, направленного на удовлетворение реальных информационных потребностей пользователя (ИПП). Эти особенности независимо от специфики решаемых пользователями задан сводятся к следующим:

1) *Изменяемость.* ИПП не может быть заранее чётко определена в спецификациях на разработку системы общения. ИПП неизбежно изменяется в ходе разработки и эксплуатации системы.

2) *Несовпадение взглядов на мир.* Представления, имеющиеся у пользователя и системы о языке общения и проблемной области, относительно которых ведётся общение, могут не совпадать. Исходя из этого, процесс общения должен предусматривать разъяснения смысла неизвестных терминов, обнаружение и устранение несовпадающих представлений, а так же предупреждение ошибочных толкований, т.е. установление общих то

чек зрения на обсуждаемые в процессе общения сущности.

3) Связность общения. Процесс общения не может быть ограничен обменом изолированными парами "вопрос-ответ", т.к. в большинстве реальных случаев ИПП не может быть выражена в виде одного вопроса (предложения). Часто требуется определить ситуацию, в которой возникла ИПП, т.е. предпослать запросу на решение некоторой задачи контекст, в котором эту задачу надо решать. Кроме того, процесс удовлетворения ИПП - решение некоторой задачи, в большинстве реальных предложений требует взаимодействия, основанного на смешанной инициативе участников. Поэтому процесс общения должен иметь сложную разветвлённую структуру и состоять из обмена связанными высказываниями.

4) *"Неправильность" высказываний пользователя.* Для выражения ИПП пользователь может применять как "правильные" предложения, т.е. такие, которые будут однозначно поняты и верно обработаны системой, так и "неправильные". Неправильности могут быть вызваны, во-первых, тем, что пользователь обычно не в состоянии учесть все ограничения системы общения в частности её возможностей и знаний; во-вторых, использованием умолчаний, характерных для естественного общения и допускающих неоднозначное толкование высказываний, и, в-третьих, отклонением предложений от грамматической нормы.

Недостатки традиционных средств общения потребовали создание средств нового поколения, которые должны быть способны настраиваться на РГПП и адаптироваться к их изменению, представлять и объяснять свою точку зрения на проблемную область, а также

учитывать точку зрения пользователя, поддерживать связный диалог и уметь обрабатывать "неправильные" высказывания. Разработка этих средств ведётся в настоящее время по двум основным направлениям:

1) Направление, развиваемое преимущественно специалистами по системам обработки данных, заключается в повышении уровня и увеличении непроцедурности формализованных языков общения.

2) Развивается в рамках ИИ и предполагает использование конечными пользователями для взаимодействия с ЭВМ естественного языка, семантически и прагматически ограниченного проблемной областью, относительно которой ведётся общение. ЕЯ-системы разработаны в рамках второго направления.

Чтобы быть полноправным участником общения ЕЯ-система должна выполнять некоторые обязательные функции. К этим функциям относятся:

1) введение диалога - определение его структуры и той роли, которую система и пользователь выполняют на текущем шаге диалога;

2) понимание - преобразование поступающих от пользователя высказываний на ЕЯ в высказывания на языке внутреннего представления;

3) обработка высказываний - формирование или определение знаний на решение задач на данном шаге диалога;

4) генерация - формирование выходных высказываний на ЕЯ.

В соответствии с высказанными функциями общения схема ЕЯ- системы может быть представлена:

1) Диалоговый компонент.

В связи с тем, что возможности существующих ЕЯ- систем не позволяют им самостоятельно формировать целесообразное поведение, в систему обычно вводится информация, определяющая общую и тематическую структуры диалога. По структуре и текущему состоянию диалога диалоговый компонент формирует (если инициатива принадлежит системе) или определяет (если инициатива принадлежит пользователю) задание, выполняемое системой на текущем шаге.

Ведение диалога выполняется по одной из двух схем: диалог ведёт пользователь, диалог ведёт система. Для системы весь диалог сводится к выработке реакций на текущие высказывания пользователя.

Вторая задача диалогового компонента вызвана тем, что реакции одного участника могут не соответствовать ожиданиям другого. Формирование перехвата происходит в тех случаях, когда система определяет, что текущая ситуация не соответствует ситуации, предусмотренной структурой диалога. Если же перехват инициативы осуществляет пользователь, то задача системы - обработать его, т.е. распознать наличие перехвата инициативы, определить новую цель (тему), на которую перешёл пользователь, и перейти на структуру диалога, соответствующую новой теме.

2) Компонент понимания высказываний предназначен для выделения смысла входного высказывания и выражения этого смысла на внутреннем языке системы. На этапе анализа выделяются описания сущностей, упомянутых во входном высказывании, выделяются свойства этих сущностей и отношения между ними. Анализ выполняется отдельным блоком-анализатором. Анализаторы различаются по ряду параметров основные из них:

- тип анализируемых предложений,
- выделяемые описания сущностей;
- глубина проникновения в смысл;
- используемые для анализа средства

Интерпретация заключается в отображении входного высказывания на знания системы. Основными задачами интерпретации являются:

- буквальная интерпретация высказывания в контексте диалога; (состоит в том, чтобы, учитывая контекст диалога, идентифицировать образы тех сущностей области интерпретации, которые имел в виду говорящий)

- интерпретация высказывания на намерения говорящего. (состоит в том, чтобы, применяя имеющиеся у системы методы вывода, определить, как обрабатываемое высказывание соотносится с целями и планами участников общения)

3) Компонент генерации высказываний решает в соответствии с результатами, полученными остальными компонентами системы, две основные задачи:

- генерация смысла является сложной и мало изученной. Тип высказывания зависит от состояния системы и результатов, полученных предыдущими компонентами. Так, если система должна генерировать ответ на вопрос, то необходимо определить по состоянию системы, будет ли ответ прямой или косвенный.

- вторая задача состоит в синтезе естественно-языкового выражения, соответствующего внутреннему представлению выходного высказывания. Данная задача подразделяется на этапы семантического, синтаксического и морфологического синтеза. Сложность задачи синтеза определяется требованиями к естественности и выразительной мощности выходных высказываний.

Для понимания принципов построения ЕЯС важен вопрос об используемых в системе знаниях. Знания ЕЯС можно классифицировать собственно знания; способ представления знаний. К основным видам знаний относятся факты и операционные знания.

Способ представления знаний включает два аспекта: способ организации знаний и модель представления.

Лекция № 5 ТЕМА: Основные положения систем речевого общения.

Основные положения

В системах искусственного интеллекта с элементами естественно-языкового (ЕЯ) общения обычно предполагается, что в качестве средства общения используется письменная речь. Это не всегда удобно, а во многих случаях и неэффективно. Использование устной речи как средства общения позволяет почти на порядок повысить скорость ввода информации, разгрузить зрение и освободить руки, осуществить речевое общение на значительном расстоянии и по телефону. Если не затрагивать общих проблем ЕЯ-общения, которые связаны с пониманием речи, с созданием преобразователей «смысл—текст» и «текст—смысл», то узко специальными проблемами, стоящими перед разработчиками систем речевого общения (СЮ), становятся проблемы создания преобразователей «текст — речевой сигнал» и «речевой сигнал — текст». Первая из них называется проблемой *синтеза речи,* вторая *— анализа и распознавания речи.*

В системах ЕЯ-общения под текстом обычно понимают *орфографический,* или буквенный (как пишется), текст, в СРО—*фонемный* (как слышится). В создании преобразователей орфографического текста в фонемный и наоборот не существует особых проблем, хотя сложность таких преобразователей для разных языков будет различной (ср. русский и английский). Поэтому применительно к СЮ можно ограничиться проблемами разработки преобразователей «цепочка фонем—речевой сигнал» и «речевой сигнал—цепочка фонем». В речи фонема выступает в двух аспектах. С одной стороны, это элементарная смыслоразличительная единица письменной речи, с другой — абстрактное обозначение конкретного звука устной речи. Отметим сразу, что не существует счетного множества, а тем более одного-единственного звука речи, соотносимого с фонемой. Если учесть, что на речевой звук налагается множество экстралингвистических факторов: индивидуальные особенности речи и голоса, эмоциональное и физиологическое состояния говорящего, электроакустические характеристики среды и тракта передачи, а также шумы, помехи и искажения, то сложность проблемы, стоящей перед создателями СРО, пожалуй, трудно переоценить.

Современные исследования в области СРО начаты в индустриально развитых странах в начале 60-х годов. Первые промышленные СРО появились в конце 70-х годов. К настоящему

времени созданы разнообразные СРО для разных сфер применения. Это связано с осознанием потенциальными потребителями преимуществ СРО:

- удобство, простота и естественность процедуры общения, требующей минимума специальной подготовки;
- возможность использования для связи с ЭВМ обычных телефонных аппаратов и существующей телефонной сети;
- устранение ручных манипуляций с одновременным увеличением скорости вода информации (в 3—5 раз по сравнению с клавиатурным вводом) и разгрузка зрения при получении информации.

Первое и второе преимущества с наибольшим эффектом проявляется в автоматизированных системах управления (АСУ) предприятия, организации или отрасли. Однако пока АСУ выполняет в основном информационные функции. С внедрением в АСУ речевой технологии общения ЭВМ станет по-настоящему активным звеном управления. Круг пользователей системы в силу простоты и естественности общения с ЭВМ неизмеримо расширится. При этом человеко-машинное взаимодействие будет осуществляться с помощью местной или городской телефонной сети на значительном расстоянии. В любое время большая группа пользователей сможет одновременно обращаться к ЭВМ за получением разного рода нормативно-справочной информации. Имея необходимую базу знаний — модель каждого конкретного производства, система сможет п0 запросу пользователя выдать необходимую консультацию по телефону в случае возникновения трудностей в той или иной сфере производства. Кроме того, система сможет выполнять функции диспетчера, обращаясь по телефону в необходимое время к конкретным работникам за получением данных о ходе производства, предупреждая их о наступлении критических ситуаций, напоминая о сроках поставки изделий или выполнения разного рода обязательств, а также осуществлять учет и контроль. Несомненно, во всех этих случаях эффективная работа СРО возможна лишь при условии, что она является составной частью систем искусственного интеллекта.

Третье свойство — разгрузка зрения и рук — важно для создания нового поколения систем оперативного человеко-машинного управления сложными объектами. К ним относятся прежде всего системы управления движением, энергетическими установками и другие АСУ ТП, САПР и АРМ.

Основой для разработки современных СРО является лингвоакустическая и информационная теории речеобразования и восприятия речи. Лингвистическая теория рассматривает фонетические и просодические характеристики речи, акустическая — акустические характеристики (признаки) фонем и просодем, информационная—структуру речевого сигнала. Эффективность СРО тем выше, чем полнее реализованы в ней принципы функционирования естественной речевой системы человека. При этом не обязательно, чтобы искусственные СРО копировали структурные особенности работы естественной системы. Важно, чтобы СРО как можно ближе была в функциональном отношении к естественной.

Фонетическая и просодическая структуры речи

Минимальной смысловой единицей речи является *фонема.* В русском языке 42 фонемы, из них 6 гласных и 36 согласных. В других языках число фонем и соотношение гласных и согласных различно. Например, в английском языке 20 гласных (из них 5 дифтонгов) и 24 согласных, во французском—16 гласных и 20 согласных.

Акустические характеристики каждой фонемы обусловлены артикуляционными особенностями ее образования: местом и способом. Под *местом образования* понимается положение сужений артикуляционного тракта в ядре фонем, определяющее его конфигурацию (артикуляционную статику) и в конечном итоге — его резонансные свойства.

Способ образования характеризует энергетические и динамические особенности артикуляции фонемы. По способу образования фонемы делятся на взрывные, аффрикаты, щелевые, дрожащие, носовые, боковые, плавные и гласные. Эти группы фонем различаются в основном, динамикой движения артикуляторов. Кроме того, взрывные,

аффрикаты и щелевые дополнительно различаются по признаку наличия-отсутствия колебании голосовых связок (звонкие-глухие).

В потоке речи в зависимости от конкретных условий акустико-артикуляцион-ные характеристики фонем изменяются, что приводит к появлению оттенков фонем — *аллофонов.* Аллофоны подразделяются на комбинаторные и позиционные. *Комбинаторные аллофоны* обусловлены соседством данной фонемы **с** другими фонемами и образуются вследствие наложения в потоке речи артикуляции одного звука на другой—так называемого эффекта коартикуляции. *Позиционные аллофоны* обусловлены положением фонемы в слове или фразе но отношению к ударному слогу, концу и началу слова и т. д.

Учет этих и других факторов позволяет дать оценку общему числу аллофонов гласных Nr=480 и согласных N_c=8880 которая, указывая порядок этого числа, наглядно иллюстрирует степень вариантности фонем в речевом потоке.

Другой класс лингвистических понятий, учет которых исключительно важен при создании высококачественных СРО, это интонация и ударение, определяющие *просодию речи.* Интонация и ударение играют важную роль при восприятии речи человеком. По ним определяются коммуникативная направленность высказывания, логический смысл, выделение главного и общего (рема и тема), осуществляется вычленение семантически связанных отрезков речи и объединение речевых элементов внутри этих отрезков. Зачастую в зависимости от интонационного оформления логически идентичные высказывания могут иметь различное семантическое значение. Физически интонация и ударение реализуются совокупностью акустических средств - просодических характеристик речи, к которым относятся:

Мелодика - изменение частоты основного тона голоса;

Ритмика - текущее изменение длительности звуков и пауз;

Энергетика - текущее изменение интенсивности звука.

Процесс чтения текста человеком предполагает наличие автоматической процедуры формирования основного тона, интенсивности звука, длительности звуков и пауз на основе анализа входного текста. Это значит, что преобразование орфографического текста в последовательность фонем должно сопровождаться выделением информации, необходимой для задания просодических характеристик речевых сигналов. Для этой цели текст анализируется и по определенным правилам разбивается на следующие основные элементы: фраза, синтагма, акцентная группа, фонетическое слово. Эти элементы маркируются соответственно фразовым, синтагматическим, групповым и словесным ударениями. Каждой синтагме присваивается один из возможных интонационных типов. Основными интонационными типами являются восклицательно-повествовательный, вопросительный. Соответствует также большое число модальных и эмоциональных оттенков этих основных интонационных типов.

Минимальным элементом, из совокупности которых складывается интонация синтагмы, фразы и текста в целом, является акцентная группа. Под *акцентной группой* понимается одно или несколько полнозначных слов синтагмы, объединенных общим просодическим (мелодическим, ритмическим и энергетическим) контуром, привязанным к единому групповому ударению. Число акцентных групп не всегда совпадает с числом фонетических слов в синтагме. Выделение слова в акцентной группе делает словесное ударение главным или основным, другие слова в акцентной группе получают слабое или не основное ударение. Основное ударение чаще всего получает первое слово акцентной группы. Это позволяет сформулировать простое правило установления границ акцентной группы: левая граница совпадает с началом слова, имеющего групповое ударение, правая - с началом слова следующей акцентной группы.

Акцентные группы ранжируется относительно границ синтагмы на конечные, начальные и серединные, которые вносят различный по значимости вклад в формирование просодического контура синтагмы. Основное разнообразие мелодических контуров

реализуется на конечной акцентной группе, существенно меньшее - на начальной и серединной.

Мелодика, ритмика и энергетика акцентной группы задаются нормированными значениями частоты, длительности и интенсивности на трех ее участках: ядре, предъядре и заядре. *Ядром* акцентной группы является ударный слог, отмеченный знаком группового ударения, *предъядром* и *заядром* — соответственно предшествующие ему и следующие за ним фонемы акцентной группы.

Как показали экспериментальные исследования просодических контуров естественной речи и опыты по восприятию интонации и ударения синтетических стимулов, интонационный тип фразы определяется положением максимума мелодического контура относительно главноударного гласного. Положению максимума в центре гласного соответствует интонация побуждения или восклицания, сдвигу его влево—интонация завершения или утверждения, а сдвигу вправо— интонация незавершенности или вопроса. Другие параметры мелодического контура—значение максимума и его ширина—более важны при передаче эмоциональных оттенков. Положение ударения в слове, синтагме или фразе определяется в большей степени длительностью ударного гласного и в меньшей его интенсивностью. Положение ударения в слове узнается в 100 % случаев, если длительность ударного гласного превышает длительность безударного более чем в 1,5 раза.

Принципы построения систем речевого общения

Классификация речевых процессоров

В общем случае *СРО* строятся на базе специализированных речевых процессоров двух основных типов: анализаторов и синтезаторов.

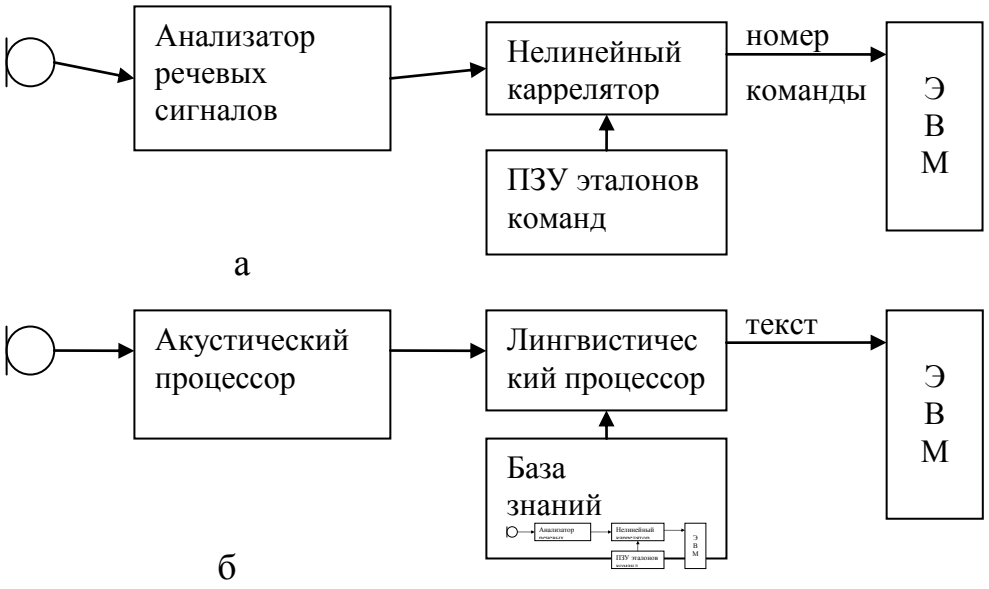
Анализаторы. Эти устройства предназначены для преобразования речевых сигналов с микрофона (информационный поток сигналов 10^5 бит/с) в последовательность цифровых кодов с существенно меньшим информационным потоком (10^4—10^1 бит/с) и с обязательным сохранением передачи смыслового компонента речи. Анализаторы подразделяются на два основных класса: анализаторы сигналов и анализаторы сообщений.

В *анализаторах сигналов* сокращение информационного потока достигается только за счет учета акустических и статистических характеристик речевого сигнала без обращения к его смысловой функции.

В *анализаторах речевых сообщений* (распознавателях) осуществляется сжатие информационного потока за счет введения операции распознавания смысловых элементов речи (фразы, слова, морфемы, фонемы). Анализаторы речевых сообщений, в свою очередь, подразделяются на две группы: анализаторы ограниченного словаря и универсальные.

Анализаторы ограниченного словаря ориентированы на распознавание заданного конкретной задачей числа речевых команд (обычно порядка 10^2), т. е. на идентификацию одной из произнесенных речевых команд словаря в виде номера команды (рис. 1,а). Распознавание осуществляется путем нелинейного во времени сопоставления эталонов команд с произносимой командой и выбора наиболее схожего эталона. В большинстве существующих анализаторов ограниченного словаря формирование эталонов осуществляется в процессе обучения на используемый словарь команд и голос диктора. Чаще всего процесс обучения состоит в однократном прочтении оператором всего словаря команд.

Рис. 1. Схема анализаторов речевых сообщений ограниченного словаря (а) и универсального (б)



Еще одним ограничением большинства современных анализаторов этого типа является требование изолированного произнесения речевых команд, т. е. с паузами между словами от 0,3 до 1 с. Распознавание слитной речи даже ограниченного словаря— пока нерешенная научная проблема.

Универсальные анализаторы ориентированы на текущее распознавание полного набора смысловых элементов речи (фонем или морфем), из которых может быть составлено и в конечном счете распознано любое слово или слитно произнесенное речевое сообщение (рис. 1,6). Распознавание осуществляется лингвистическим процессором по правилам, заложенным в базе знаний.

Синтезаторы речи. Эти устройства предназначены для преобразования кодовой последовательности, поступающей от ЭВМ, ПЗУ или линии связи, в непрерывный речевой сигнал. Синтезаторы подразделяются на классы и группы по тем же признакам, что и анализаторы речи. Классам анализаторов речевых сигналов и сообщений соответствуют синтезаторы речевых сигналов и сообщений

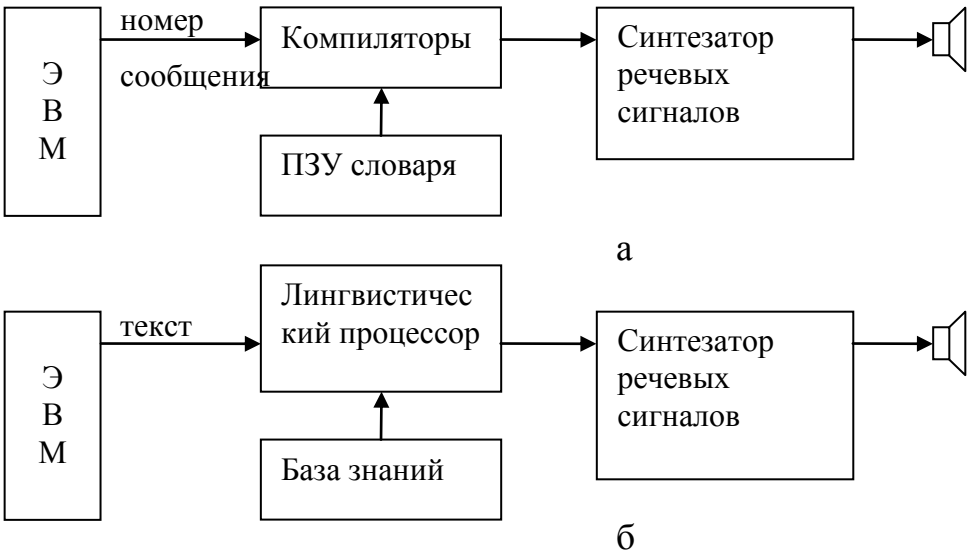
Синтезаторы сообщений делятся по аналогичным признакам на синтезаторы ограниченного словаря — компиляторы и универсальные.

В *компиляторах* (рис. 2,а) любое сложное речевое сообщение может быть получено путем компиляции (простого соединения) элементов речи. Элементы речи начитываются диктором, соответствующие им сигналы компрессируются тем или иным способом, кодируются и записываются в ПЗУ. При синтезе речевого сообщения из ПЗУ закодированные речевые элементы считываются в нужной последовательности и одновременно восстанавливается речевой сигнал. Очевидной простотой компиляционного метода и его технической реализации объясняется большое количество сообщений о проектируемых и законченных разработках компиляторов к их применении.

Удовлетворительный по качеству компиляционный синтез речи возможен лишь при использовании в качестве элементов речи отдельных фраз либо словоформ, подставляемых в определенное место стандартной фразы. Попытки добиться высококачественного синтеза произвольного текста простой компиляцией словоформ, слогов или аллофонов не привели к положительным результатам. Все эти элементы речи тесно связаны внутри фразы. В слитной речи не существует аналогов этих элементов, произнесенных изолированно, и наоборот, речь из изолированно произнесенных элементов звучит ненатурально. В связи с этим подготовка словаря в компиляторах представляет собой самостоятельную и сложную проблему. Для успешного ее решение в каждом конкретном случае применения синтезатора многие фирмы за рубежом идут на создание специализированных центров

проектирования речи, оснащенных соответствующим оборудованием и персоналом лингвистов-прикладников.

Рис. 2. Схемы синтезаторов речевых сообщений ограниченного словаря (а) и универсального (б)



При разработке *универсальных синтезаторов* речевых сообщений стремятся получить функциональную модель речеобразования, адекватную реально существующим языковым и акустическим явлениям. На входе такой модели орфографический или фонемный текст произвольного содержания, на выходе— звучащая речь (рис. 2,6). По своему существу разрабатываемые в рамках данного подхода синтезаторы являются кибернетической функциональной моделью чтения текста человеком. В базе знаний синтезатора хранится не только информация об элементарных единицах речи (эталоны фонем и интонем), но и алгоритмические правила их модификации в зависимости от конкретного контекста звуковой реализации. Процесс применение этих правил к эталонам фонем и интонем для входного синтезируемого текста реализуется лингвистическим процессором. К настоящему времени качество речи и стоимость универсальных синтезаторов достигли коммерчески приемлемых показателей, и они начинают оказывать серьезную конкуренцию компиляторам в силу простоты применения, малого расхода памяти на элемент речи, неограниченности состава словаря синтезируемых сообщений.

Лекция № 6 Тема: Назначение, классификация и область применения систем обработки визуальной информации

Назначение, классификация и области применения

В различных областях науки и техники ощущается рост потребностей в переработке, анализе и отображении визуальной, информации. В этом направлении принято выделять три основных типа задач: собственно обработку изображений, когда и исходные данные, и результаты обработки представляются в изобразительной форме; анализ (интерпретацию, распознавание или "понимание") изображений, когда входные данные являются изображением, а результат представляется в неизобразительной форме, например в виде текстового описания наблюдаемой сцены; синтез изображений (машинную графику), когда на входе имеется описание (алгоритм построения) изображения, а на выходе но нему строится само изображение. Взаимосвязь трех перечисленных типов задач показана на рис. 1.

Обработка изображений связана с преобразованием изобразительной информации вновь в изобразительную форму. Примером здесь может служить устранение искажений или

дефектов на изображении, улучшение качества получаемой визуальной информации путем повышения контраста, подчеркивание контуров объектов и т. д.

Задачей анализа изображений является получение из изображения, поданного на вход системы, неизобразительного описания. Это описание может быть различных уровней общности— от простого указания номера или имени класса, к которому относится анализируемое изображение, до подробной характеристики наблюдаемой сцены с указанием отдельных объектов и отношений между ними. В последнем случае обычно говорят о «понимании» изображений. Иногда называют анализ изображений распознаванием, хотя, этот термин имеет более узкое значение, относясь преимущественно к идентификации отдельных объектов на изображениях. К типичным задачам анализа можно отнести распознавание рукописных или печатных знаков, дешифрацию аэрофотоснимков, анализ наблюдаемых сцен.

Синтез изображений обычно отождествляют с машинной графикой. В настоящее время синтезируются на ЭВМ не только «графические» картины, наоборот, синтез все больше претендует на создание полноцветных реалистических изображений по их описаниям в неизобразительной форме. Сюда относятся и системы имитации визуальной обстановки на тренажерах, и системы геометрического моделирования в САПР, и системы компьютерного киноискусства. Наряду с этим сохраняются и традиционные приложения машинной графики: вывод информации на экран дисплея в виде диаграмм, графиков; построение чертежей и т.д.

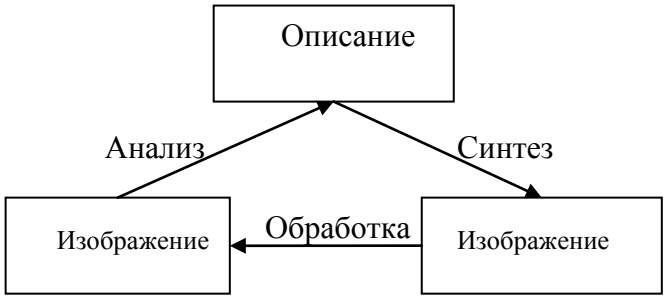


Рис. 1. Взаимосвязь направлений цифровой обработки изображений

Интерактивной графикой называют направление, предполагающее не только синтез изображения машиной, но и ввод пользователем в систему данных в графической форме, например, с помощью светового пера или манипулятора типа "мышь".

Системы обработки и анализа видеоданных, которые обычно работают со входной информацией в виде полутоновых или многоспектральных изображений, в нашей стране обычно объединяют под названием *автоматизированных систем обработки изображений* (АСОИз). В настоящее время системы, решающие достаточно сложные и «интеллектуальные» задачи анализа изображений, принято выделять в отдельный класс систем анализа изображений (САИ). Системы, задачами которых являются синтез изображений и обработка графической информации, называют *системами машинной графики* (СМГ); употребляется также термин «системы обработки графической информации» (СОГИ). Следует отметить, что такое деление достаточно условно, и за последнее время наметилась устойчивая тенденция к сближению этих систем. Все чаше АСОИз оперируют графической информацией, решая в том числе и задачи синтеза, а СМГ применяются для построения полутоновых изображений с привлечением различных методов их обработки.

Степень «интеллектуализации» АСОИз и систем машинной графики может быть различной в зависимости от уровня представления и использования в этих системах знаний специалистов по обработке изображений и специалистов предметных областей. Она колеблется от возможности применения набора утилит (операций) обработки, анализа или синтеза изображений, которые в грубом приближении можно интерпретировать как

записанные в процедурной форме записи специалистов, до поддержки специальных баз знаний в развитых экспертных системах анализа изображений.

Рассмотрим системы трех классов:

- традиционные «мало интеллектуальные» АСОИз, без понимания состава, функций и особенностей построения которых трудно представить себе уровень развития интеллектуальных систем;
- системы анализа видеоданных, основанные на знаниях, которые образуют специальный класс экспертных систем анализа изображений;
- интеллектуальные системы машинной графики.

Автоматизированные системы обработки изображений

Одно из направлений наиболее широкого практического применения АСОИз— обработка результатов дистанционных исследований. Среди многочисленных развивающихся прикладных областей здесь можно упомянуть: геологию (исследование и поиск природных и ископаемых ресурсов); сельское и лесное хозяйство (обнаружение аномалий, слежение за состоянием участков, предсказание урожая); анализ состояния окружающей среды и климата; метеорологию (предсказание погоды, анализ воздушных течений и температурных полей); гидрологию и океанографию (наблюдение за водными ресурсами, течениями, распределениями температуры, планктона, льдов); картографию (составление фотографических и других карт, накопление картографических данных, территориальный анализ); слежение за объектами на земной поверхности (транспорт, судовождение и т. д.).

Быстро развивается направление обработки биомедицинских данных. Здесь АСОИз находят применение в микробиология и цитологии (анализ снимков с микроскопов, анализы крови, хромосомный анализ), рентгенологии, ультразвукоскопии, инфраскопии (обработка двух - и трехмерных изображений внутренних органов, анализ распределения температур тела, томография), сравнительных исследованиях (идентификация отпечатков пальцев, идентификация состояний больных). Недостаточно развито пока применение АСОИз в производственных системах, однако потребности автоматизации управления и производства должны привести к быстрому росту числа применений АСОИз в таких областях, как материаловедение (обнаружение дефектов, анализ и предсказание свойств материалов, структурный анализ), сборочные работы (системы технического зрения для роботов и гибких автоматизированных производств), системы ориентации и анализа обстановки в реальном времени (для различных видов транспорта, летательных и космических аппаратов).

Исследовательские АСОИз создаются обычно как «универсальные», не привязанные жестко ни к какому классу видеоданных. В то же время данные конкретных предметных областей имеют свою специфику и особенности, что необходимо требует проблемной ориентации АСОИз, разрабатываемых для прикладных применений. Первые такие системы были специализированными, предназначенными для решения конкретных узких задач с помощью ограниченного набора алгоритмов. С развитием методов обработки изображений, удешевлением аппаратуры, появлением новых технологий (и первую очередь СБИС) и технических средств (микропроцессоров, интеллектуальных видеотерминалов, памяти большой емкости и т. п.) проявилась устойчивая тенденция к переходу от систем специализированных к системам проблемно-ориентированным, призванным решать некоторый класс задач в рамках данной предметной области.

Проблемно-ориентированные прикладные АСОИз используют, как правило, достаточно богатый арсенал различных методов и средств обработки данных, анализируют различную входную информацию, активно используют возможности человека. Поэтому в таких системах должны быть функциональные различные блоки

и подсистемы, обеспечивающие не только решение собственно задач обработки изображений, но и ввод и обработку других типов данных и знаний, накопление, хранение и отображение результатов, организацию взаимодействия с пользователем и другими системами. Наличие разнородных компонентов, работающих совместно для достижения общей цели, есть признаки превращения современных АСОИз в *интегрированные системы.*

Внутренняя интеграция затрагивает различные компоненты АСОИз: средства и методы обработки, сами обрабатываемые данные.

Интеграция средств обработки означает объединение в системе различных универсальных и/или специализированных устройств, имеющих возможность параллельной работы над данными.

Интеграция методов обработки означает наличие в системе функциональных подсистем и обрабатывающих алгоритмов для решения широкого круга задач и достижения результатов различными путями (возможно, параллельными). При этом к результату и приходят различными доступными путями, применяя разные методы к одним и тем же данным и интегрируя (на основе опыта и интуиции исследователя или заложенных в систему знаний) получаемую выходную информацию.

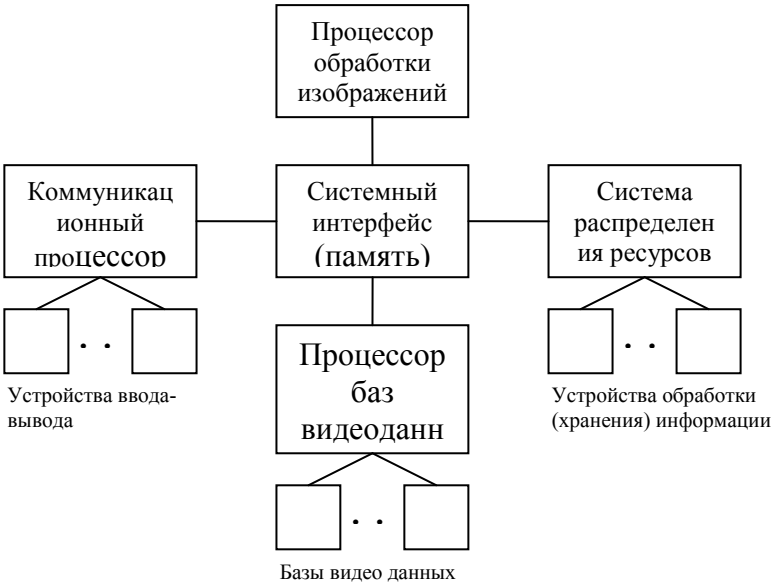


Рис.2 Структура интегрированной АСОИз

Работа по анализу изображения становится своего рода многократным экспериментированием с исходными данными. Характерной чертой такого эксперимента является совместное применение методов обработки изображений, математической статистики, искусственного интеллекта, анализа данных и других, причем в обработку по возможности вводятся все доступные данные, включая промежуточныерезультаты. Появляется необходимость организации эффективного обмена данными в системе, создания развитых средств хранения и обеспечения доступа к данным (включая базы видеоданных). В настоящее время это уже характерно для многих АСОИз, применяемых в дистанционных исследованиях.

Ключевым звеном АСОИз обычно считается процессор(ы), производящий обработку изображении. В первых* системах в качестве обрабатывающего процессора часто применялись крупные универсальные ЭВМ, работающие в режиме разделения времени. Это решение оказалось малоэффективным, поскольку и системы команд, и операционные системы таких ЭВМ не ориентированы на задачи обработки изображении. В архитектуре АСОИз 70-х годов наряду с появлением специализированных процессоров обработки изображений получило широкое распространение «монопольное использование средних и малых ЭВМ, ориентация которых на решение задач видеоданных

достигалась разработкой соответствующего программного обеспечения. В настоящее время широкое распространение получили так называемые рабочие станции обработки изображений и системы, построенные на базе персональных ЭВМ.

Наличие одного процессора редко удовлетворяет требованиям по производительности системы, особенно при больших потоках обрабатываемых данных. Крупные АСОИз строятся на основе специальных многопроцессорных конфигураций для параллельной обработки элементов одного или нескольких изображений, при этом упомянутые средние или малые ЭВМ обычно переходят в разряд управляющих.

Другими архитектурными компонентами АСОИз являются:

- устройства ввода видеоданных в систему (ТВ-камеры, дальномеры, малокадровые сканеры, фототелеграфные аппараты и т. д.);
- устройства отображения (ТВ-мониторы, растровые и векторные дисплеи, устройства вывода на твердые носители - бумагу и фотопленку, графопостроители);
- запоминающие устройства (на гибких и жестких магнитных дисках, оптических дисках; устройства расширения оперативной памяти для хранения изображений).

В настоящее время в интегрированных системах аппаратно реализуются не только параллельные процессоры обработки изображений, но и другие подсистемы, ранее традиционно реализуемые программным путем. Прежде всего это средства доступа к различным видам памяти и средства, поддерживающие принятие решений. Особенно большую пользу приносят такие реализации при решений задач распознавания образов, интерпретации видеоданных, анализа сцен. Фактически аппаратные средства здесь поддерживают функционирование базы данных и базы знаний, но на качественно ином уровне. Основные решаемые задачи для базы видеоданных — это поиск эталонных объектов или их отдельных частей (видеопризнаков), поиск необходимых изображений по их идентификаторам. Чтобы решить эти задачи, разрабатываются специализированные процессоры баз видеоданных и массивы ассоциативной памяти с возможностью контекстного доступа к информации (адресация по содержимому).

Важное направление развития программного обеспечения АСОИз—создание эффективных систем хранения и поиска видеоданных (баз и банков видеоданных). Потребность в базах видеоданных ощущается в области дистанционных исследований, поэтому именно здесь можно проследить тенденции развития программного обеспечения этих систем.

Все чаще в состав программного обеспечения АСОИз включаются и средства ведения баз знаний. Пока это преимущественно специализированные подсистемы для использования базы знаний о конкретной предметной области (например, о поиске месторождений определенного типа на аэрофотоснимке), но в скором времени можно ожидать появления инструментальных средств для разработки и наполнения знаниями экспертных систем анализа визуальной информации.

Системы анализа изображений

Выделяют три основных класса задач, решаемых САИ (системы анализа изображения), которые определяют их различное назначение:

1. Распознавание объектов. В системах этого типа решаются задачи локализации определенных объектов в поле изображения, измерения заданных параметров этих объектов, принятия решения по полученным результатам измерений. Примерами могут служить системы медицинской диагностики, где необходимо локализовать участки изображения, указывающие на отклонение от нормы, определить степень отклонения; системы контроля протекания процессов (например, сгорания топлива в камере котельной); системы локализации дефектов печатных плат и др.
2. Содержательная интерпретация изображений. К системам, в которых решается эта задача, относится большинство САИ. Сюда входят системы управления движением робота по поступающей видеоинформации, интерпретации аэрофотоснимков, анализа снимков и

сопоставления их с картой местности и др. В отличие от систем первого типа в данном случае задача заключается в полной содержательной интерпретации снимка, «понимании» отношений, связывающих объекты реального мира, представленные своими проекциями на двухмерном изображении.

3. Получение справочной информации в архивах видеоданных. Задачей этих систем является выдача информации пользователю о наличии объектов определенного типа на изображениях, хранящихся в архиве видеоданных, его параметрах, отношений к другим объектам и т. п. Запрос может быть представлен как текстовое описание объекта или как видео образ. К системам такого типа относятся различные информационно-поисковые системы в картографии, метеорологии, криминалистике и т. п.

Для решения указанных задач САИ должны обеспечивать выполнение ряда функций, среди которых основными являются следующие.

1. Хранение и поиск формализованных знаний экспертов предметной области и обработки изображений, формализованными знаниями являются установленные факты, закономерности, количественные соотношения, видеоданные (эталоны, прототипы объектов), а также правила, определяющие последовательность и состав операции над данными для достижения заданных целей (например, для получения проекции трехмерного тела на плоскость).
2. Выполнение операций обработки изображений (контрастирование, устранение шума, выделение контурных линий и т. п.).
3. Выполнение операций логических выводов на основе формализованных знаний и результатов обработки видеоданных, построение и проверка гипотез.
4. Анализ промежуточных результатов работы системы и принятие решений о дальнейших действиях, невозможности решения задачи, необходимости дополнительной информации и др.
5. Анализ запросов пользователя, поддержка диалога, документирование действий системы и формирование пояснений пользователю.
6. Пополнение баз знаний и видеоданных системы новыми сведениями, данными.

Конкретная САИ не обязательно выполняет все перечисленные функции. Современные системы анализа изображений довольно разнообразны по своей структуре и организации взаимодействия ее составных частей.

На рис. 3 приведена упрощенная структурная схема ЭСАИ в ее наиболее полном варианте, соответствующая системе содержательной интерпретации изображений.

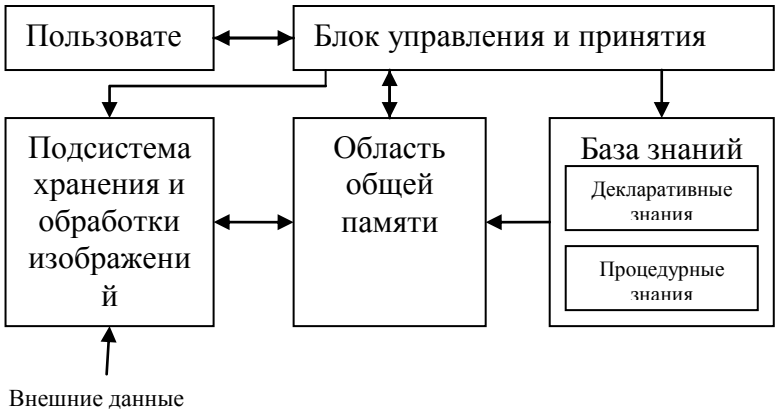


Рис. 3. Структура экспертной системы анализа изображений

Подсистема хранения и обработки изображений содержит программные и аппаратные модули для выполнения различных операций над видеоданными, поиска изображений по видео- и текстовому запросу, а также архив изображений. Важной особенностью подсистемы является модульный принцип построения, благодаря которому система может собирать технологические цепочки последовательностей

операций для решения конкретных задач. Решению этой задачи могут предшествовать операции поиска заданного изображения в базе видеоданных.

В базе знаний системы хранятся формализованные знания специалистов предметной области и обработки изображений. Условно в ней можно выделить два компонента: декларативные и процедурные знания.

Декларативные знания отражают сведения об объектах, их составе, взаимном расположении, характере появления, описывают отношения между объектами и т. п. Выделяют три типа описания объектов: контекстное, визуальное и характеризующее повторяемость объекта в сцене. Следует отметить, что для большинства ЭСАИ существенная часть декларативных знаний связана с пространственными, количественными, классификационными отношениями объектов, а такие отношения, как временные, причинно-следственные, пока еще слабо используются,

Особенно следует отметить отношение типа обобщение-спецификация, используемое в большинстве современных ЭСАИ и отражающее иерархию абстракций - последовательности представлений объектов от примитивов (перепадов яркостей, однородных участков и т. п.) до абстрактных понятий (здание, дорога, автомобиль и т. п.). Построение иерархий абстракций является ключевым моментом воспроизведения процесса восприятия. Считается, что окончательная интерпретация изображения должна выполняться системой на верхних понятийных уровнях иерархии.

Второй основной частью базы знаний являются процедурные знания, показывающие, какие действия, в какой последовательности нужно предпринимать системе при выполнении определенных условий, отражающих сложившуюся ситуацию анализа данных, формализованных знаний и результатов обработки изображений. Процедурные знания используются для выполнения двух функций: пополнения описания сложившейся ситуации новыми фактами из базы знаний; построения гипотез и выбора конкретных действий по преобразованию полученных результатов и переходу к другому состоянию решения задачи.

Процедурные знания обычно представляются в виде правил, разделенных на две части: условие выполнения действия и назначаемое действие (последовательность действий) в случае выполнения условия. Интерпретация объектов может считаться состоятельной, если выполняются все предусмотренные между ними отношения, и т. д.

Область общей памяти, называемая также доской объявлений (blackboard) или базой данных, является частью системы, обеспечивающей передачу результатов работы всех подсистем друг другу. Обычно область общей памяти представляет собой реляционную базу данных, куда в ходе решения задачи записываются промежуточные интерпретации участков изображений (карты, платы интерпретации), установленные факты, количественные соотношения и другие результаты. В область общей памяти могут заноситься также пояснения действий ЭСАИ, с тем чтобы пользователь мог при необходимости оценить рациональность поведения системы, выявить причину принятия того или иного решения.

Блок управляющих процедур и принятия решений является координирующим центром системы, анализирующим запросы пользователя, сложившуюся ситуацию анализа, определяющим контекст и стратегию решения. Информация о промежуточных результатах анализа поступает в блок из области общей памяти. По результатам анализа состояния области общей памяти (сложившейся ситуации) блок управляющих процедур и принятия решений определяет состоятельность полученной интерпретации, генерирует гипотезы (о принадлежности объектов определенным классам, о возможности нахождения других объектов и т. п.) и «поручает» другим модулям системы выполнение конкретных действий. Окончательное решение также принимается этой подсистемой.

Процесс решения задачи содержательной интерпретации изображения является многоступенчатым, итеративным. На начальной стадии обычно выполняется предварительная сегментация изображения.

Следующим этапом является получение начальной интерпретации результатов предварительной сегментации. Это включает в себя измерения определенных параметров (например, площадей или периметров полученных областей с одинаковой яркостью) и генерацию гипотез (с помощью сопоставления результатов измерений с информацией из базы знаний).

Все результаты интерпретации записываются в область общей памяти. Изменения состояния этой области означают, что процесс интерпретации не закончен. Блок управляющих процедур и принятия решений анализирует состояние области и вырабатывает новую последовательность действий. Процесс решения задачи обычно считается завершенным, если не происходят существенные изменения в области общей памяти (на карте интерпретации).

Обычно в ЭСАИ используются две глобальные стратегии решения задач интерпретации изображений (а также их комбинации): управление целью (называемое также управлением сверху вниз или обратным выводом) и управление данными (называемое также управлением снизу вверх или прямым выводом). В первом случае последовательность действии выбирается исходя из поставленной цели. Такая стратегия соответствует продвижению сверху вниз по иерархии абстракций: от понятия объекта к его геометрическим примитивам на изображении. Вторая стратегия соответствует движению в обратном направлении по иерархии абстракций. Последовательность действий определяется на основе полученных результатов обработки данных.

Большинство разработчиков ЭСАИ придерживаются мнения, что обе стратегии должны присутствовать в решении задачи системой. Переключение стратегий остается нерешенной проблемой, так же как и нет единого мнения о том, в каких случаях система должна выбирать ту или иную стратегию,

Системы машинной графики

Основными областями применения систем машинной графики являются:

- автоматизация проектирования (построение чертежей, проекций, планов,двух- и трехмерных визуальных моделей деталей и объектов);
- дизайнерские работы в таких областях, как автомобиле- и самолетостроение, архитектура, пошив одежды;
- картография (составление, корректировка, хранение и генерация карт); автоматизация исследований (отображение в визуальной форме результатов экспериментов, вычерчивание графиков, гистограмм, диаграмм и т. д.);
- организация эффективного взаимодействия человека с ЭВМ (графические языки, интерактивная графика);
- синтез картин и сцен (для тренажеров самолето- и автовождения, в машинной живописи, в компьютерных играх);
- синтез динамических картин (компьютерная мультипликация и кинематография).

Машинная графика стала обязательной частью любой интеллектуальной системы, реализованной на ЭВМ. Это связано прежде всего с тем, что скорость переработки человеком информации, представленной в графической форме, выше по сравнению со скоростью переработки алфавитно-цифрового набора данных. Особенно заметен этот эффект на этапе восприятия информации, так как при этом максимально активизируется ассоциативное мышление человека. Наиболее ярко преимущества графического представления данных проявляются в интерактивных системах машинной графики, подразумевающих активную обратную связь с пользователем.

Состав типичной СМГ представлен на рис. 4. Схематически работу системы можно представить следующим образом. На основании базы знаний и банка данных ЭВМ генерирует модель «мира». Затем эта модель преобразуется программой синтеза изображения в удобную для восприятия человеком форму. Сформированная таким образом информация выдается на устройство отображения графической информации, вызывая на

нем появление соответствующего изображения, которое может быть задокументированно с помощью устройства получения твердых копий. На этапе окончательного формирования изображения в него могут включаться готовые фрагменты из банка видеоданных. Пользователь через устройство ввода информации может формировать и дополнять базу знаний и банк данных, корректировать модель «мира» и программу синтеза изображения, а также производить непосредственные операции над изображением (вырезание и перемещение фрагментов, их склейку, масштабирование, раскраску и т. д.), воздействуя через сервисные программы непосредственно на буфер устройства отображения.

Модель «мира» является центральным звеном практически всех современных СМГ.

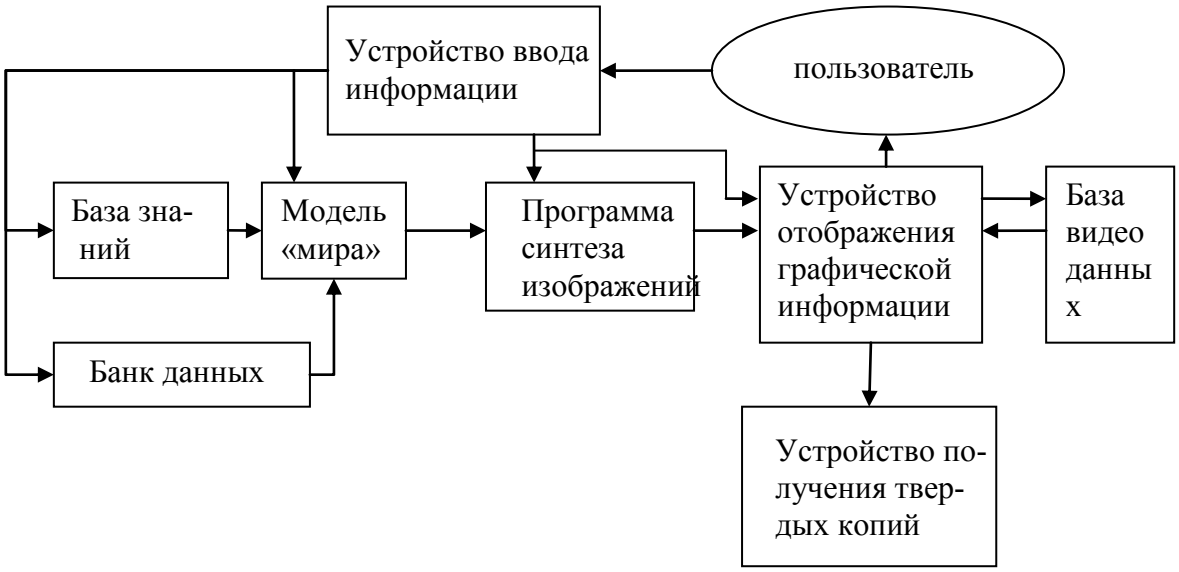


Рис. 4. Структура системы машинной графики

По способу описание объектов модели «мира» можно разделить на модели, задающие объект в виде набора данных и в виде операции его построения. Сама операция построения может быть представлена в процедурной форме или задаваться структурой данных. Способ описания, использующий полностью детерминированное задание всех исходных данных, характеризующих объект, основывается на концепции имитации процесса фотографирования реальной сцены. Это приводит к необходимости однозначного задания всех объектов модели «мира» в момент ее «фотографирования» воображаемым фотоаппаратом. Сейчас чаще используется задание объектов «мира» в виде операции их построения. В этом случае объект синтезируется непосредственно в процессе построения. Это значительно облегчает задачу разработчика модели, так как вместо детального описания объекта он задает лишь закон его построения (например, в виде закона распределения вероятности реализации конкретных фрагментов объекта). Кроме геометрического описания объектов в модели «мира» хранятся и другие параметры, необходимые для синтеза изображения (например, цвета поверхностей объектов).

При создании модели «мира» используются база знаний и банк данных. База знаний вводит фактически законы и ограничения, действующие при создании объектов «мира». Банк данных позволяет «записывать» модель «мира» стандартными данными, необходимыми для ее работы. Банк данных и база знаний могут постоянно изменяться и дополняться пользователем в процессе эксплуатации системы.

Сформированную таким образом модель «мира» специальная программа преобразует в массив чисел, соответствующий изображению. Наиболее сложной задачей такого рода является синтез реалистичного изображения трехмерной сцены. Большой прогресс в этой области был достигнут к началу 80-х годов. К настоящему времени в

процессе синтеза цветного трехмерного реалистичного изображения можно выделить следующие этапы:

- удаление скрытых поверхностей, т. е. поверхностей или их частей, которые не видны из точки нахождения воображаемого наблюдателя. Этот этап может быть выполнен, например, с помощью «алгоритма Z-буфера» или «алгоритма художника»;
- закраска видимых поверхностей с учетом свойств самой поверхности (цвет, фактура, отражательная способность), ее относительного местоположения, ориентации, а также свойств источников света и других поверхностей;
- выравнивание интенсивностей соседних поверхностей с помощью алгоритмов закраски Гуро или Фонга.

В последнее время благодаря быстрому удешевлению аппаратной части СМГ возрос интерес к синтезу динамических изображений в реальном масштабе времени, т. е, к так называемой компьютерной мультипликации. Динамическая графика наводит применение в компьютерных играх, тренажерных системах, АСНИ. При синтезе динамических объектов, движущихся по законам модели «мира», программа синтеза изображения кроме перечисленных выше операций выполняет также ряд геометрических преобразований, соответствующих моделируемому движению.

Банк видеоданных служит для воспроизведения готовых изображений и их фрагментов. Он может постоянно пополняться новыми данными, полученными как извне, так и в процессе функционирования самой системы.

В качестве базовых для синтеза изображения могут использоваться универсальные ЭВМ практически всех классов — от персональных компьютеров до суперЭВМ. Простейшими графическими средствами снабжены и современные бытовые компьютеры. Для синтеза реальных трехмерных сцен необходимы 32-разрядные ЭВМ.

Разработка графических программ на языках высокого уровня началась в конце 60-х годов. При этом широко использовалась идеология «виртуальных» устройств, обеспечивающая независимость программного обеспечения высокого уровня от особенностей внешних устройств конкретной вычислительной системы. Программист пользовался лишь координатами виртуального экрана, а программы нижнего уровня автоматически переводили их в координаты реального экрана.

Приблизительно в этот же период времени были разработаны два пакета графических программ, доступные для всех типов дисплеев. Это трехмерная система Core Graphics Svstem и двухмерная система Graphical Kernel System, принятая Международной организацией по стандартизации. Однако у этих пакетов есть существенный недостаток. Они разрабатывались еще до того, как стали отдавать предпочтение растровым дисплеям, поэтому при использовании этих пакетов невозможно в полной мере учитывать преимущества пикселей.

Лекция № 7 Тема: Назначение систем машинного перевода.

Назначение машинного перевода

Машинный перевод (МП), или автоматический перевод (АП),—интенсивно развивающаяся область научных исследований, экспериментальных разработок и уже функционирующих систем (СМП), в которых к процессу перевода с одного естественного языка (ЕЯ) на другой привлекается ЭВМ. СМП открывают быстрый и систематический доступ к информации на иностранном языке, обеспечивают оперативность и единообразие в переводе больших потоков текстов, в основном научно-технических. Работающие в промышленном масштабе СМП опираются на большие терминологические банки данных и, как правило требуют привлечения человека в качестве пред-, интер- или постредактора. Современные СМП, в особенности те, которые опираются при переводе на базы знаний в определенной предметной области, относят к классу систем искусственного интеллекта (ИИ).

Основные сферы использования МЦ

1. В отраслевых службах информации при наличии большого массива или постоянного потока иноязычных источников. Если СМП используются для выдачи сигнальной информации, постредактирование не требуется.

2. В крупных международных организациях, имеющих дело с многоязычным политематическим массивом документов. Таковы условия работы в Комиссии Европейских сообществ в Брюсселе, где вся документация должна появляться одновременно на девяти рабочих языках. Поскольку требования к переводу здесь высоки, МП нуждается в постредактировании.

3. В службах, осуществляющих перевод технической документации, сопровождающей экспортируемую продукцию. Переводчики не справляются в требуемые сроки с обширной документацией (так, спецификации к самолетам и другим сложным объектам могут занимать до 10000 и более страниц). Структура и язык технической документации достаточно стандартны, что облегчает МП и даже делает его предпочтительным перед ручным переводом, так как гарантирует единый стиль всего массива. Поскольку перевод спецификаций должен быть полным и точным, продукция МП нуждается в постредактировании.

4. Для синхронного или почти синхронного перевода некоторого постоянного потока однотипных сообщений. Таков поток метеосводок в Канаде, который должен появляться одновременно на английском и французском языках.

Помимо практической потребности делового мира в СМП, существуют и чисто научные стимулы к развитию МП: стабильно работающие экспериментальные системы МП являются опытным полем для проверки различных аспектов общей теории понимания, речевого общения, преобразования информации, а также для создания новых, более эффективных моделей самого МП.

С точки зрения масштаба и степени разработанности СМП можно разбить на три основных класса: промышленные, развивающиеся и экспериментальные.

Лингвистическое обеспечение систем машинного перевода

Процесс МП представляет собой последовательность преобразований, применяемых к входному тексту и превращающих его в текст на выходном языке, который должен максимально воссоздавать смысл и, как правило, структуру исходного текста, но уже средствами выходного языка. К лингвистическому обеспечению СМП относится весь комплекс собственно лингвистических, металингвистических и так называемых «экстралингвистических» знаний, которые используются при таком преобразовании.

В классических СМП, осуществляющих непрямой перевод по отдельным предложениям (пофразный перевод), каждое предложение проходит последовательность преобразований, состоящую из трех частей (этапов): анализ —> трансфер (межъязыковые операции)—>синтез. В свою очередь, каждый из этих этапов представляет собой достаточно сложную систему промежуточных преобразований.

Цель этапа анализа построить структурное описание (промежуточное представление, внутреннее представление) входного предложения, | Задача этапа трансфера (собственно перевода)—преобразовать структуру входного предложения во внутреннюю структуру выходного предложения. К этому этапу относятся и замены лексем входного языка их переводными эквивалентами (лексические межъязыковые преобразования). Цель этапа синтеза—на основе полученной в результате анализа структуры построить правильное предложение выходного языка.

Лингвистическое обеспечение стандартной современной СМП включает:

1) словари;
2) грамматики;
3) формализованные промежуточные представления единиц анализа на разных этапах преобразований.

Помимо стандартных, в отдельных СМП могут иметься и некоторые нестандартные компоненты. Так, экспертные знания о ПО могут задаваться с помощью специальных концептуальных сетей, а не в виде словарей и грамматик.

Механизмы (алгоритмы, процедуры) оперирования с имеющимися словарями, грамматиками и структурными представлениями относят к математико-алгоритмическому обеспечению СМП.

Одно из необходимых требований к современным СМП—высокая модульность. С лингвистически содержательной точки зрения это означает, что анализ и следующие за ним процессы строятся с учетом теории лингвистических уровней. В практике создания СМП различают такие уровни анализа:

- досинтаксический анализ (в него входит морфологический анализ - МорфАн, анализ оборотов, неопознанных элементов текста и др.);

- синтаксический анализ СинАн (строит синтаксическое представление предложения, или СинП); в его пределах может выделяться ряд подуровней, обеспечивающих анализ разных типов синтаксических единиц;

- семантический анализ СемАн, или логико-семантический анализ (строит аргументно-предикатную структуру высказываний или другой вид семантического представления предложения и текст);

- концептуальный анализ (анализ в терминах концептуальных структур,отражающих семантику ПО). Этот уровень анализа используется в СМП, ориентированных на очень ограниченные ПО. По сути дела, концептуальная структура является проекцией схем ПО на лингвистические структуры, часто даже не на семантические, а на синтаксические. Только для очень узких ПО и ограниченных классов текстов концептуальная структура совпадает с семантической; в общем случае полного совпадения не должно быть, так как текст подробнее любых концептуальных схем.

Синтез теоретически проходит те же уровни, что и анализ, но в обратном направлении. В работающих системах обычно реализован только путь от СинП до цепочки слов выходного предложения.

Лингвистическое разграничение разных уровней может проявляться также в разграничении используемых в соответствующих описаниях формальных средств (набор этих средств задается для каждою уровня отдельно). На практике часто задаются отдельно лингвистические средства МорфАн и совмещаются средства СинАн и СемАн. Но разграничение уровней может оставаться только содержательным при использовании в их описаниях единого формализма, пригодного для представления информации всех выделяемых уровней.

С технической точки зрения модульность лингвистического обеспечения означает отделение структурного представления фраз и текстов (как текущих, временных знаний о тексте) от «постоянных» знаний о языке, а также языковых знаний • от знаний ПО; отделение словарей от грамматик, грамматик - от алгоритмов их обработки, алгоритмов « от программ. Конкретные соотношения различных модулей системы (словари— грамматики, грамматики — алгоритмы, алгоритмы — программы, декларативные — процедурные знания и др.), включая распределение лингвистических данных по уровням,— это то основное, что определяет специфику СМП.

Словари. Словари анализа, как правило, одноязычные. Они должны содержать всю информацию, необходимую для включения данной лексической единицы (ЛЕ) в структурное представление. Часто разделяют словари основ (с морфолого-синтаксической информацией: часть речи, тип словоизменения, подкласс, характеризующий синтаксическое поведение ЛЕ и т. п.) и словари словозначений, содержащие семантическую и концептуальную информацию: семантический класс ЛЕ, семантические надежи (валентности), условия их реализации во фразе и т. д.

Во многих системах разделены словари общеупотребительной и терминологической лексики. Такое разделение дает возможность при переходе к текстам другой предметной области ограничиваться лишь сменой терминологических словарей. Словари сложных ЛЕ (оборотов, конструкций) образуют обычно отдельный массив, словарная информация в них указывает на способ «собирания» такой единицы при анализе. Часть словарной информации может задаваться в процедурной форме, например, многозначным словам могут сопоставляться алгоритмы разрешения соответствующего типа неоднозначности. Новые виды организации словарной информации для целей МП предлагают так называемые «лексические базы знаний». Наличие разнородной информации о слове (называемой лексическим универсумом слова) приближает такой словарь, скорее к энциклопедии, чем к традиционным лингвистическим словарям.

Грамматики и алгоритмы. Грамматика и словарь задают лингвистическую модель, образуя основную часть лингвистических данных. Алгоритмы их обработки, 1. е. соотнесения с текстовыми единицами, относят к математико-алгоритмическому обеспечению системы.

Разделение грамматик и алгоритмов важно в практическом смысле тем, что позволяет менять правила грамматики, не меняя алгоритмов (и соответственно программ), работающих с грамматиками. Но далеко не всегда такое разделение возможно. Так, для системы с процедурным заданием грамматики и тем более С процедурным представлением словарной информации такое разделение нерелевантно. Алгоритмы принятия решений в случае недостаточной (неполнота входных данных) или избыточной (вариантность анализа) информации в больший мере эмпиричны, их формулировка требует лингвистической интуиции. Задание общего управляющего алгоритма, ведающего порядком вызова разных грамматик (если их несколько в одной системе), также требует лингвистического обоснования. Тем не менее существующая тенденция — отделить грамматики от алгоритмов так, чтобы все лингвистически содержательные сведения задавались в статической форме грамматик, а алгоритмы сделать настолько абстрактными, что они смогут вызывать и обрабатывать разные лингвистические модели.

Наиболее четко разделение грамматик и алгоритмов наблюдается в системах, работающих с контекстно-свободными грамматиками (КСГ), где модель языка— грамматика с конечным числом состояний, а алгоритм должен обеспечить для произвольно взятого предложения дерево его вывода по правилам грамматики, и если таких выводов несколько, то перечислить их. Такой алгоритм, представляющий собой формальную (в математическом смысле) систему, называется анализатором. Описание грамматики служит для анализатора, обладающее универсальностью, таким же входом, как и анализируемое предложение. Анализаторы строятся для классов грамматик, хотя учет специфических особенностей грамматики может повысить эффективность анализатора.

Грамматики синтаксического уровня — наиболее разработанная часть и с точки зрения лингвистики, и с точки зрения их обеспечения формализмами.

Основные типы грамматик и реализующих их алгоритмов:

- цепочечная грамматика фиксирует порядок следования элементов, т. е, линейные структуры предложения, задавая их в терминах грамматических классов слов (артикль+существительное+предлог} или в терминах функциональных элементов (подлежащее+сказуемое);

- грамматика составляющих (или грамматика непосредственно составляющих— НСГ) фиксирует лингвистическую информацию о группировке грамматических элементов, например, именная группа (состоит из существительного, артикля, прилагательного и других модификаторов), предложная группа (состоит из предлога и именной группы) и т. д. до уровня предложения. Грамматика строится как набор правил подстановки, или исчисление продукций вида А—»В...С. НСГ представляют собой грамматики порождающего типа и могут использоваться как при анализе, так и при синтезе: предложения языка порождаются многократным применением

таких правил;

- грамматика зависимостей (ГЗ) задает иерархию отношений элементов предложения (главное слово определяет форму зависимых). Анализатор в ГЗ основан на идентификации хозяев и их зависимых (слуг). Главным в предложении является глагол в личной форме, так как он определяет число и характер зависимых существительных. Стратегия анализа в ГЗ—сверху вниз: сначала идентифицируются хозяева, затем слуги, или снизу вверх: хозяева определяются процессом подстановки;

- категориальная грамматика Бар-Хиллела—это версия грамматики составляющих, в ней только две категории — предложения S и имени п. Остальные определяются в терминах способности комбинироваться е этими главными в структуре НС. Так, переходный глагол определен как n\S, так как он сочетается с именем и слева от него, образуя предложение S.

Существует много способов учета контекстных условий: грамматики метаморфозы и их варианты. Все они являются расширениями КС-правил. В общем виде это значит, что правила продукций, переписываются так: А [а]—>В[Ь], ... ,С [с], где малыми буквами обозначены условия, тесты, инструкции и т. д., расширяющие исходные жесткие правила и дающие грамматике гибкость и эффективность.

В грамматике обобщенных составляющих—ТСС введены метаправила, являющиеся обобщением закономерностей правил КС1.

В грамматиках расширенных сетей переходов—РСП предусмотрены тесты и условия к дугам, а также инструкции, которые надо выполнить в случае, если анализ пошел по данной дуге. В разных модификациях РСП дугам может приписываться вес, тогда анализатор может выбирать путь с наибольшим весом. Условия могут разбиваться на две части: контекстно-свободные и контекстно-зависимые.

Разновидностью РСПГ являются каскадные РСПГ. Каскад—это РСП, снабженная действием 1шшш1. Это действие вызывает остановку процесса в данном каскаде, запоминание информации о текущей конфигурации в стеке и переход к более глубокому каскаду с последующим возвратом в исходное состояние. РСП обладает рядом возможностей трансформационных грамматик. Она может использоваться и как генерирующая система.

Метод анализа с помощью граф-схемы позволяет сохранить частичные результаты и представить варианты анализа.

Новым и сразу завоевавшим популярность методом грамматического описания является лексшо-фунщиональная грамматика (ЛФГ). Она устраняет необходимость трансформационных правил. Хотя ЛФГ основывается на КСГ, проверочные условия в ней отделены от правил подстановки и «решаются» как автономные уравнения.

Унификационные грамматики (УГ) представляют собой следующий после граф-схем этап обобщения модели анализа: они способны воплощать грамматики различных видов. УГ содержит четыре компонента: пакет унификации, интерпретатор для правил и лексических описаний, программы обработки направленных графов, анализатор с помощью граф-схемы. УГ объединяют грамматические правила со словарными описаниями, синтаксические валентности с семантическими.

Центральной проблемой любой системы анализа ЕЯ является проблема выбора вариантов. Для ее решения грамматики синтаксического уровня дополняются вспомогательными грамматиками и методами разбора сложных ситуаций. В НС-грамматиках применяют фильтровый и эвристический методы. Фильтровый метод состоит в том. что сначала получают все варианты анализа предложения, а затем отбраковывают те, которые не удовлетворяют некоторой системе условий-фильтров. Эвристический метод с самого начала строит лишь часть вариантов, более правдоподобных с точки зрения заданных критериев. Использование весов для отбора вариантов является примером применения эвристических методов в анализе.

Семантический уровень гораздо меньше обеспечен теорией и практическими разработками. Традиционной задачей семантики считается снятие неоднозначности

синтаксического анализа - структурной и лексической. Для этого используется аппарат селективных ограничений, который привязан к рамкам предложений, т. е. вписывается в синтаксическую модель.

Наиболее распространенный тип СемАн основан на так называемых падежных грамматиках. В основе грамматики—понятие глубинного, или семантического, падежа. Падежная рамка глагола является расширением понятия валентность: это набор смысловых отношений, которые могут (обязательно или факультативно) сопровождать глагол и его вариации в тексте. В пределах одного языка один и тот же глубинный падеж реализуется разными поверхностными предложно — падежными формами. Глубинные падежи в принципе позволяют выходить за рамки предложения, а выход в текст означает переход к семантическому уровню анализа.

Поскольку семантическая информация в отличие от синтаксической, опирающейся в первую очередь на грамматики, сосредоточена в основном в словарях, в 80-е годы интенсивно разрабатываются грамматики, позволяющие «лексикализовать» КСГ. Ведется разработка грамматик, основанных на исследовании свойств дискурса.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ЛАБОРАТОРНЫМ РАБОТАМ

ЛАБОРАТОРНАЯ РАБОТА № 1.
ИЗУЧЕНИЕ РАБОТЫ С ИНТЕГРИРОВАННОЙ ОБОЛОЧКОЙ СИСТЕМЫ ТУРБО ПРОЛОГ.

Общие сведения

1. Турбо-Пролог, версия 2.0

Система Турбо Пролог версия 2.0 может работать на ПЭВМ, совместимых с IBM PC XT/AT и PS/2, с ОЗУ минимум 384 Кбайт и двумя НГМД по 360 Кбайт. Рекомендуется иметь ОЗУ 512/640 Кбайт и НМД типа "Винчестер". В файле config.sys должно быть указано files=20 buffers=40.

Программа на Турбо Прологе состоит из следующих в определенном порядке секций и имеет следующую структуру [2]:

```
constants    /* Секция объявления констант.  Может отсутствовать */
domains      /* Секция объявления нестандартных и/или составных типов данных.  Может отсутствовать  */
database - имя_ВБД   /* Необязательная секция объявления предикатов для работы с внутренней базой данных (ВБД) */
predicates   /* Секция  объявления  предикатов */
clauses      /* Секция объявления правил и фактов */
goal         /* Секция объявления внутренней цели. Может отсутствовать */
```

При составлении программы на Прологе необходимо соблюдать следующие ограничения:

- комментарии в программе могут располагаться в программе на любом месте. Комментарий начинается либо с символа % либо с последовательности символов /* и заканчиваться */;
- в программе может использоваться только один раз секция GOAL;
- все предикаты в CLAUSES с одинаковыми именами должны записываться подряд;
- большинство стандартных предикатов выполняют несколько функций в зависимости от состояния параметров, входящих в предикат. Известные параметры называют входными (INPUT – (i)), неизвестные – выходными (OUTPUT – (o)). Совокупность входных параметров определяет работу предиката. Эта совокупность называется проточным шаблоном.

1.1. Интегрированная оболочка системы Турбо-Пролог предоставляет следующие возможности:

- создавать и редактировать тексты программ;
- выполнять и отлаживать программы;

- транслировать программы в объектные файлы;
- компоновать объектные файлы в исполняемые модули;
- получать справочную информацию, изменять размеры окон и их цвет;
- устанавливать параметры и конфигурацию системы.

При первоначальном входе в интегрированную среду Турбо-Пролога на экране монитора появляется главное меню (рис.1). В верхней строке находятся названия 6 основных режимов работы системы. Текущее положение в меню отмечено выделяющейся по цвету и яркости прямоугольной полоской. Перемещая эту полоску (курсор) с помощью клавиш с горизонтальными стрелками нажатием клавиши Enter можно выбрать необходимый режим. Это можно сделать также одновременным нажатием клавиши Alt и первой буквы названия соответствующего меню, например, для выбора режима редактирования достаточно нажать Alt-E.

Для удобства работы для наиболее часто используемых операций в оболочке Турбо Пролога вместо выбора из меню (или подменю) можно использовать нажатие функциональных клавиш, либо определенного сочетания клавиш (Hot keys). Действие той или иной функциональной клавиши может быть различным в зависимости от того, в каком режиме находится система. Более полную подсказку можно получить нажатием клавиш Alt-H.

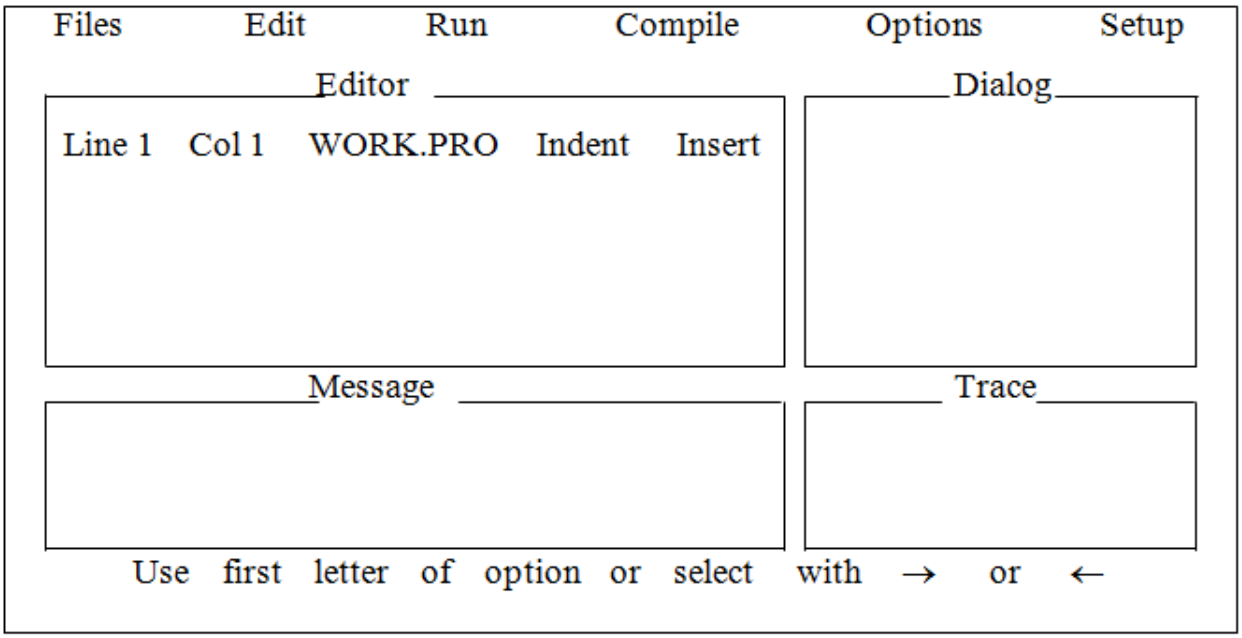


Рис. 1

Экран разделен на 4 окна:

- окно редактора текстов, в которое загружаются отлаживаемые или редактируемые программы;
- окно диалога является по умолчанию окном, если в программе не назначены другие, ввода и вывода из программы;
- окно для вывода сообщений системы;
- окно сообщений отладчика (трассировки).

Первоначальное разделение экрана может быть изменено в режиме Setup. Окно, в котором находится курсор, называется текущим окном.

Нижняя строка является строкой подсказки или иначе навигационной строкой. Навигационную строку вверху, показывающую местоположение курсора (Line - номер текущей строки, Col - позиция курсора в строке), название редактируемого файла и режимы редактирования, имеет также окно редактирования. Если имя редактируемого фала не задано, то по умолчанию он называется WORK.PRO. Если файл с таким именем находится в текущей директории, то он автоматически загружается в окно редактирования.

Все режимы главного меню, кроме Edit и Run, содержат дополнительные подменю. Выход из любого подменю осуществляется нажатием клавиши Esc.

1.2. Цикл разработки программы

Турбо Пролог поддерживает в широком смысле структурное программирование и сопровождение проекта. Цикл разработки программы (без постановочной части) можно представить следующей схемой:

1. С помощью встроенного редактора текстов исходный текст программы вводится в систему.
2. Периодически, по мере ввода новых предикатов, программа транслируется для выявления синтаксических ошибок, которые тут же устраняются.
3. С помощью встроенных средств трассировки производится отладка каждого нового предиката.
4. Периодически производится структуризация программы. Независимые части программы выделяются в отдельные модули.
5. Процесс повторяется до получения программы, удовлетворяющей внешним спецификациям, которые, в частности, сами могут быть написаны на Турбо Прологе.

При выборе режима Files в верхней части экрана под словом Files появится подменю (рис. 2.).

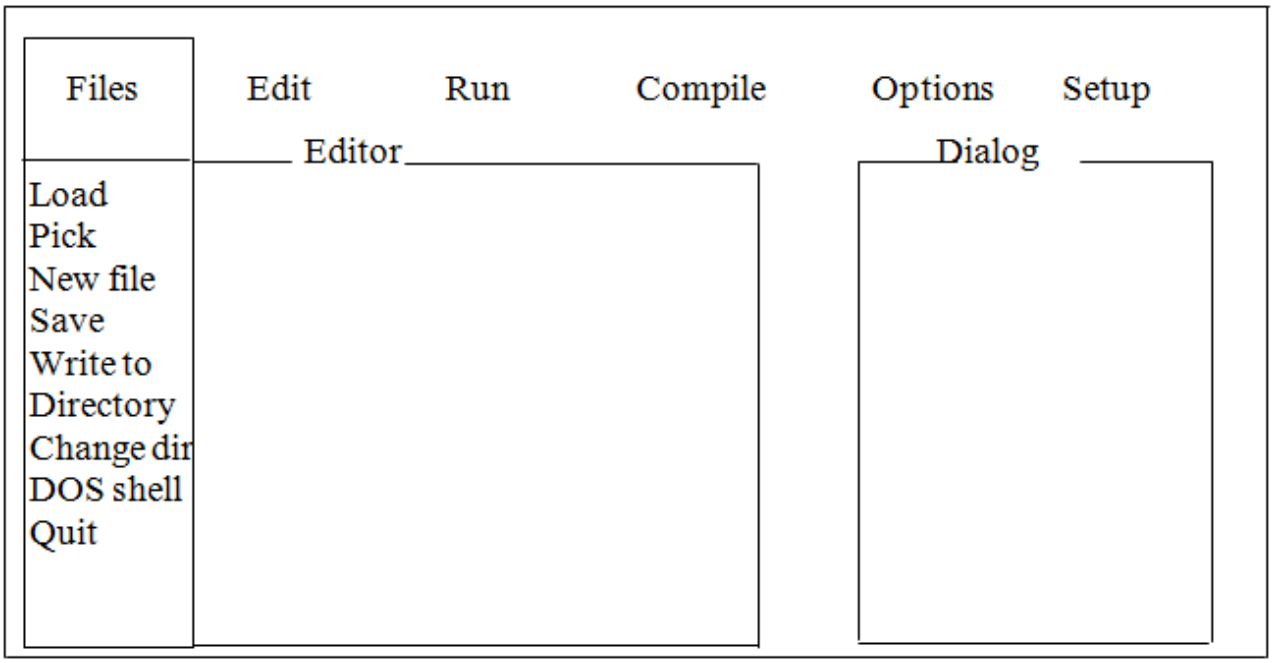


Рис. 2.

2. Основные режимы работы

Основные режимы работы заданы главным меню в верхней строке экрана. Такими режимами являются:

Files - работа с файлами и взаимодействие с MS-DOS
Edit - ввод и редактирование программы
Run - выполнение и отладка программы
Compile - трансляция и компоновка программы
Options - установка режимов компиляции
Setup - установка режимов работы оболочки

2.1. Работа с файлами и взаимодействие с MS-DOS

При выборе режима Files в верхней части экрана под словом Files появится подменю (рис. 2.).

Перемещение по подменю осуществляется также с помощью клавиш со стрелками, вводом первой (заглавной) буквы слова. Для быстрого выполнения наиболее частых операций имеются зарезервированные функциональные клавиши или одновременное нажатие нескольких клавиш (Hot keys).

Load - позволяет загрузить исходный текст программы на Прологе в окно редактора для его последующей модификации, выполнения или отладки.

Pick - позволяет работать из оболочки Турбо-Пролога одновременно с 8-ю файлами.

New file - очистка окна редактора текстов для ввода нового файла.

Save - сохранение редактируемой программы в файле на диске.

Write to - запись редактируемой программы в заданный файл на диске.

Directory - вывод содержимого указанной директории на экран.

Change dir - смена текущей директории. Равносильна команде ChDir в MS DOS.

OS shell - выполнение команды операционной системы. Выход из системы Турбо-Пролог в MS-DOS с возвратом по команде EXIT.

Quit - окончание работы. Выход из системы Турбо-Пролог в MS-DOS.

При работе с файлами приняты следующие соглашения об их типах:

.PRO - файл с исходным текстом программы

.BAK - файл сохранения после редактирования

.PRJ - файл с именами модулей, входящих в проект разрабатываемой системы

.EXE - исполняемый файл после компоновки

.OBJ - объектный файл после трансляции

.ARC - упакованный файл архивации

2.2. Ввод и редактирование программы

Встроенный редактор интегрированной оболочки Турбо-Пролога имеет набор команд, который является подмножеством команд широко известного редактора текстов WordStar. Кроме того, при редактировании можно пользоваться более короткими командами, пользуясь функциональными и управляющими клавишами.

Все команды редактирования можно переопределить, однако учитывая, что они совпадают с оболочками языков Турбо Паскаль, Турбо Си и Турбо Бейсик, это не рекомендуется делать.

Как и при работе с обычным текстовым редактором, ввод может происходить в режиме вставки текста или в режиме наложения его на уже существующий. Переход с одного режима в другой осуществляется нажатием клавиши Ins.

В случае слияния файлов, копирования блока из одного файла в другой и т.п. в системе Турбо Пролог может быть открыто дополнительное окно редактирования (Aux Editor) в нижнем правом углу экрана.

Команды редактора [2] условно делятся на команды перемещения, команды удаления текста, команды работы с блоками текста и вспомогательные. Выход из редактора в главное меню - F10 или по одновременному нажатию клавиши Alt и клавиши с первой буквой имени соответствующего режима.

Если при редактировании программы возникли какие-либо трудности, то нажатие клавиши F1 вызовет следующее меню системы подсказки:

Edit Help - [Помощь при редактировании] Заголовок меню

Show help file - вывод на экран файла справочника по всей системе. Нажатие F5 позволяет увеличить окно со справочником до размеров экрана. Повторное нажатие уменьшит его до прежних размеров. Выход из справочника по Esc.

Cursor movement - вывод на экран справочника по командам для перемещения курсора.

Insert & Delete - справочник по операциям вставки и удаления текста.

Block Functions - справочник по операциям над блоками текста.

WordStar-like - справочник по командам редактирования, аналогичным командам редактора текстов WordStar.

Miscellaneous - справочник по командам редактирования не входящим в какую-либо группу.

Global functions - справочник по командам, выполняющим операции над всем текстом.

Hot keys - справочник по клавишам для быстрого выполнения наиболее часто встречающихся операций. При этом выбранная с помощью клавиш со стрелками операция может быть тут же выполнена.

Команды перемещения

Направление перемещения	Клавиши
Курсор вправо на один символ	Стрелка вправо или Ctrl-D
Курсор влево на один символ	Стрелка влево или Ctrl-S
Курсор вверх на одну строку	Стрелка вверх или Ctrl-E
Курсор вниз на одну строку	Стрелка вниз или Ctrl-X
Курсор вправо на одно слово	Ctrl-Стрелка вправо
Курсор влево на одно слово	Ctrl-Стрелка влево
Курсор в конец строки	End
Курсор в начало строки	Home
Курсор в начало окна	Ctrl-Home
Курсор в конец окна	Ctrl-End
Курсор вверх на один экран	PgUp или Ctrl-R
Курсор вниз на один экран	PgDn или Ctrl-C
Курсор в начало файла	Ctrl-PgUp или Ctrl-Q R
Курсор в конец файла	Ctrl-PgDn или Ctrl-Q C

Команды удаления текста

Название команды	Клавиши
Удаление символа над курсором	Del
Удаление символа слева от курсора	BackSpace или Ctrl-H
Удаление слова над курсором	Ctrl-T
Удаление строки над курсором	Ctrl-Y
Удаление начала строки до курсора	Ctrl-Q T
Удаление от курсора до конца строки	Ctrl-Q Y

2.3. Выполнение и отладка программ

Режим Run обеспечивает компиляцию находящейся в окне редактора текстов программы и ее выполнение. Этот режим обеспечивает быструю интерактивную отладку программы, так как при обнаружении синтаксических и семантических ошибок имеется возможность их тут же исправить. Мощным инструментом отладки программ в системе являются встроенные средства трассировки.

Возможны следующие виды трассировки:

- пошаговая трассировка всей программы
- трассировка заданных предикатов
- трассировка в режиме оптимизации
- интерактивное включение/выключение трассировки
- интерактивный вывод результатов трассировки и выходных сообщений на печать и в дисковый файл.

Отметим, что трассировка является также важным средством изучения Турбо Пролога, так как, в случае непонимания работы какого-либо предиката, его работу можно посмотреть по шагам.

По умолчанию режим трассировки выключен (Off). Для установки/отключения режима трассировки следует выборать подменю "Директивы компилятора" меню Options (См. п. 2.5) и в нем выбрать вход Trace. На экране появится небольшое меню, содержащее три входа:

Trace - включить режим полной трассировки;
ShortTrace - включить режим сокращенной трассировки;
Off - выключить трассировку.

Если выбрать Trace (выход из всех подменю по Esc), то после запуска программы на выполнение (Run или Alt-R) в окне диалога появится подсказка Goal:. После ввода цели, в окне трассировки за словом CALL: появляется вызываемый предикат.

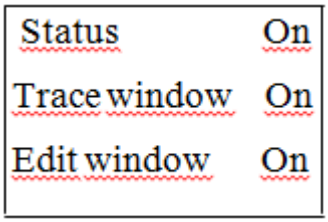
Нажатие клавиши F10 разрешает выполнение следующего шага программы. Одновременно в окне редактирования курсор показывает на выполняемый в текущий момент предикат.

После подсказки REDO: отображаются результаты вычисления (унификации).

При успешном поиске подстановке после слова RETURN: выводится ее результат. При неудаче выводится подсказка FAIL.

Трассировка в любой момент может быть прекращена нажатием клавиши Esc.

Нажатие Alt-T позволяет изменить режим трассировки. При этом на экране появляется следующее меню:



Запрет/разрешение отображения статуса

Запрет/разрешение окна трассировки

Запрет/разрешение окна редактирования

После нажатия Shift-F10 можно по желанию изменить размеры окон.

Внутри программы можно размещать передикат trace(on), который включает режим трассировки, или trace(off) - выключающий ее.

Нажатие клавиш Alt-P позволяет перенаправить вывод результатов трассировки на печать или в файл PROLOG.LOG на диске.

Режим ShortTrace отличается от Trace только тем, что компилятор производит оптимизацию программы (в режиме Trace оптимизацияне производится). Это особенно важно при отладке рекурсивных вызовов, особенно при "хвостовой рекурсии", когда рекурсивный вызов является последней подцелью в теле предиката. Этот вид рекурсивных вызовов оптимизируется Турбо Прологом и преобразуется в цикл, что экономит память при исполнении.

2.4. Трансляция и компоновка программ

Подменю режима Compile содержит следующие входы:

Memory - отладочная трансляция в память
OBJ file - трансляция в файл типа .OBJ
EXE file (auto link) - трансляция с созданием исполняемого (.EXE) файла с автоматической компоновкой модулей

Project (all modules) -трансляция всех входящих в проект модулей;

Link only - выполнение компоновки модулей в исполняемый файл.

2.5. Установка режимов компиляции

Исходный текст программы на Турбо Прологе перед выполнением проходит обязательную фазу компиляции и компоновки. Для определения параметров этих процессов служит режим Options.

Меню режима Options содержит следующие входы:

Link options (Опции компоновки)

Содержит подменю:

Map file ON/OFF - создавать или нет файл с картой компоновки.

Libraries - ввод имен пользовательских библиотек для компоновки.

Edit PRJ file (Редактирование файла с описанием проекта)

Запрашивается имя файла, в котором перечислены модули, входящие в проект. Этот файл затем используется компоновщиком.

Compiler Directive (Директивы компиляции)

Содержит подменю (названия входов даны в круглых скобках) позволяющие установить:

- распределение памяти (Memory allocation) под код, стек, тип и рекурсию. Размер этих областей устанавливается в параграфах (параграф равен 16 байтам);

- какие виды контроля выполнять во время исполнения программы (Run-time check): нажатие клавиши Breack, нарушение границ стека и целочисленное переполнение;

- допустимый уровень ошибок трансляции (Error level): ошибки недопустимы, по умолчанию (1), максимальный (2);

- выдачу предупреждения при наличии недетерминированных предикатов (Non-determ warning ON/OFF);

- предупреждение о наличии переменных, которые используются в предикате только один раз (Variable used once warning ON/OFF);

- уровень трассировки (Trace): полная, сокращенная и трассировка выключена;

- включение вывода диагностики по результатам трансляции (Diagnostics ON/OFF).

2.6. Получение справочной информации, изменение размеров окон и их цвета

Подменю режима SetUp содержит следующие входы:

Colors - изменение цвета окон при наличии цветного монитора. На экран выводится таблица цветовой гаммы, из которой для каждого окна выбирается цвет фона и цвет изображения;

Window size - изменение размеров окон. При выборе этой опции в нижней строке экрана появляется подсказка, как с помощью клавиш со стрелками изменить размер или местоположение окна.

Directories - установка директорий по умолчанию.

Miscellaneous - разные опции связанные с настройкой на адаптер видеомонитора;

Load SYS file - загрузка файла с параметрами интегрированной среды;

Save SYS file сохранение параметров настройка среды интегрированной среды в файле типа *.SYS.

3. Стандартные предикаты

3.1. Общесистемные стандартные предикаты

В этом разделе приведены предикаты [2], позволяющие использовать возможности предоставляемые операционной системой MS DOS.

- date(Год,Месяц,День) (integer, integer, integer): прототип (o,o,o) - считывает системную дату, прототип: (i,i,i) - установить дату.

Например:

date(J,M,T) - результат: J=2001,M=11,T=01

date(2001,02,26) - системная дата устанавливается на 26.02.2001.

time(Час, Минуты, Секунды, Сотые_секунды) (integer, integer, integer, integer): прототип (o,o,o,o) - связывает соответствующие параметры с текущим временем, прототип (i,i,i,i) - устанавливает системное время.

Например:

time(S,M,Sek,_) - результат: S=15,M=35,Sek=22

time(17,05,0,0) - часы будут поставлены на 17:05:00.

- system(Строка_с_командой_DOS) (string): прототип (i) - заданная строка должна быть допустимой командой DOS. Разрешаются встроенные и внешние команды DOS, такие как файлы с расширением ".BAT". Заданная команда выполняется. Предикат system не выполняется, если команда некорректна с точки зрения DOS. Если "Строка_с_командой_DOS" пустая (""), то COMMAND.COM активируется в интерактивном режиме.

Например:

system("dir A:") - выводится каталог накопителя A.

- comline(Строка) (string): прототип (o) - читает параметры, которые заданы в команде вызова скомпилированной программы на Турбо – Прологе.

Например:

TEST abc (вызов в MS - DOS) - если в программе с именем TEST стоит команда comline(X), то строка abc будет связана с X.

- beep - производит звуковой сигнал

- bios(Номер_прерывания,Регистры_входные,Регистры_выходные) (integer, reg, reg): прототип (i,i,o) - обеспечивает формирование прерывания с заданным номером. Регистры получают значения, установленные параметром "Регистры_входные". После обработки прерывания содержимое регистров связывается с параметром "Регистры_выходные". Параметры "Регистры_входные" и "Регистры_выходные" должны принадлежать домену regdom, который определяется следующим образом:

regdom = reg(AX, BX, CX, DX, SI, DI, DS, ES), где все аргументы имеют тип integer.

Например: bios($21, reg(AX,0,0,0,0,0,0,0),reg(Nr,_,_,_,_,_,_,_)) - производится чтение с текущего накопителя. Номер накопителя связывается с Nr.

3.2. Предикаты преобразования типов

- char_int(Символ,Число) (char,integer): прототип (i,o) - связывает параметр "Число" с кодом ASCII параметра "Символ"; прототип (o,i) - связывает параметр "Символ" с символом, код которого определяется параметром "Число"; прототип (i,i) - выполняется успешно, если код, определяемый параметром "Число", является ASCII – кодом символа, определяемого параметром "Символ".

Например:

char_int('A',X) - переменная X принимает значение 65.

char_int(X,66) - переменная X принимает значение 'B'.

char_int('A',65) - выполняется успешно.

- str_char(Строка,Символ) (string,char): прототип: (i,o) - заданная первым параметром строка, состоящая из единственного символа, преобразуется в символ. Символ связывается со вторым параметром; прототип (o,i) - преобразуется символ в строку, состоящую из единственного символа и связывает ее с заданной переменной; прототип: (i,i) - выполняется успешно, если параметры связаны с представлениями одного и того же символа.

Например:

str_char("A",X) - результат X='A'.

str_char(X,'A') - результат X="A".

str_char("A",'A') - выполняется успешно.

- str_int(Строка,Целое число) (string,integer): прототип:(i,o) - преобразует строку в целое число. Число связывается со вторым параметром; прототип (o,i) - связывает с первым параметром строку, представляющую собой запись целого числа, связанного со вторым

параметром; прототип(i,i) - выполняется успешно, если связанная с первым параметром строка является представлением числа, связанного со вторым параметром.

Например:

str_int(“123”,X) - результат: X=123

str_int(X,456) - результат: X=”456”

str_int(“234”,234) - выполняется успешно.

- str_real(Строка,Действительное число) (string,real): прототип:(i,o) - связывает второй параметр с действительным числом, определяемым записью числа в строке, заданной первым параметром; прототип (o,i) - связывает с первым параметром строку, представляющую собой запись действительного числа, заданного вторым параметром; прототип:(i,i) - выполняется успешно, если связанная с первым параметром строка является представлением числа, связанного со вторым параметром.

Например:

str_real(“1.23”,X) - результат: X=1.23

str_real(X,0.56) - результат: X=”0.56”

str_real(“4.567”,4.567) - выполняется успешно.

- file_str(Имя_ файла_ DOS,Строка) (string,string): прототип (i,o) - читает строку из заданного файла и связывает ее с параметром "Строка". Максимально допустимый размер строки – 64 К. Признаком конца строки является символ Ctrl – Z (десятичный код ASCII=26).

Например: file_str(“B:TEXT1”,X) - символы из файла TEXT1 на накопителе B будут прочитаны и связаны с переменной X.

- field_str(Строка, Столбец, Длина, Строка_символов) (integer, integer, integer, string): прототип (i,i,i,i) - записывает строку, связанную с параметром "Строка_символов", в поле, определяемое длинной и номерами строки и столбца. Если строка длиннее, чем заданное поле, то записывается только начало строки. Если строка короче, то оставшиеся позиции поля заполняются пробелами; прототип (i,i,i,o) - строка, определяемая длиной и позицией, связывается с параметром "Строка_символов". Проследите, чтобы поле в текущем окне соответствовало параметрам.

Например:

field_str(15,5,5,"hollo") - строка "hollo" записывается в поле, начинающееся с позиции (15,5).

field_str(10,30,5,X) - строка длиной 5, начинающаяся с позиции (10,30), связывается с переменной X.

- str_len(Строка,Длина) (string,integer): прототип (i,o) - с параметром "Длина" связывается количество символов в заданной строке; прототип (i,i) - выполняется успешно, если строка имеет заданную длину.

Например:

str_len("hollo", X) - результат: X=5.

str_len("book", 4) - выполняется успешно.

- isname(Строка) (string) прототип (i) - выполняется успешно, если последовательность символов, связанная с параметром, представляет собой имя, допустимое в Турбо – Прологе.

Например:

isname ("abcd") - выполняется успешно.

- upper_lower(Строка1,Строка2) (string, string): прототип (i,i) - выполняется успешно, если с первым и вторым параметром связаны идентичные строки, представленные соответственно прописными и строчными буквами; прототип (i,o) - связывает со вторым параметром строку, полученную из строки, связанной с первым параметром, заменой прописных букв на строчные; прототип (o,i) - связывает с первым параметром строку, полученную из строки, связанной со вторым параметром, заменой строчных букв на прописные.

Например:

upper_lower("A","a") - выполняется успешно.

upper_lower("ZDF",X) - результат: X="zdf"
upper_lower(X,"house") - результат: X="HOUSE"

3.2. Арифметические операции

+ сложение
- вычитание
* умножение
/ деление
mod абсолютная величина
div целочисленное деление

3.3. Операторы отношений

Операторы отношений являются инфиксными операторами (т.е. должны находится между двумя сравниваемыми величинами). Свободные переменные в операторах отношений не допускаются. Для операторов отношений приняты следующие обозначения:

< меньше; > больше; = равно; <= меньше или равно; >= больше равно; <> не равно.

3.4. Математические функции

Функция	**Описание**
abs(X) /* (Var) (i) */	Возвращает абсолютное значение X
round(X) /* (Var) (i) */	Возвращает округленное целое значение X
sqrt(X) /* (Var) (i) */	Возвращает квадратный корень из X
trunc(X) /* (Var) (i) */	Возвращает целое значение X отбрасывая дробную часть
exp(X) /* (Var) (i) */	Возвращает значение e в степени X
log(X) /* (Var) (i) */	Возвращает десятичный логарифм X
ln(X) /* (Var) (i) */	Возвращает натуральный логарифм X
arctan(X) /* (Radians) (i) */	Возвращает арктангенс X
cos(X) /* (Radians) (i) */	Возвращает косинус X
sin(X) /* (Radians) (i) */	Возвращает синус X
tan(X) /* (Radians) (i) */	Возвращает тангенс X

Все тригонометрические функции требуют, чтобы аргумент X задавался в радианах.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 1

Тема: Создание отношений в Прологе.

Последовательность действий

1. Изучить пояснения к работе.
2. Загрузить Турбо Пролог. На экране появится главное меню и окна, изображенные на рис. 1.
3. Нажимая клавишу с левой или правой горизонтальной стрелкой последовательно пройти по главному меню. Обратите внимание на появление подменю по мере перехода от одного режима к другому. Войти в подменю Files. С помощью клавиш с вертикальными стрелками переместить прямоугольную полоску на слово Directory. Нажать клавишу Enter (/вв/). На экран будет выведено содержание текущей директории.
4. Сделать текущим окно редактирования. Вернуться в главное меню нажатием клавиши F10.
5. Войдите в режим редактирования (Alt-E), изучите действие клавиши F1 (меню подсказок) и введите следующие определения:

```
domains                 /вв/
  fakt = symbol         /вв/
predicates              /вв/
  male(fakt)            /вв/
  female(fakt)          /вв/
  mother(fakt,fakt)     /вв/
```

```
    father(fakt,fakt)       /вв/
    wife(fakt,fakt)         /вв/
  clauses                          /вв/
    male(“саша”).                  /вв/
    female(“марина”).              /вв/
    mother(“наташа”,”саша”).  /вв/
    father(“петя”,”коля”).         /вв/
    wife(“наташа”,”петя”).         /вв/
    ……
```

6. Введите в секцию clauses 5-7 фактов для предикатов male, female, mother, father, wife.

7. Пользуясь средствами Турбо Пролога постройте предикаты для выражения следующих связей между объектами:

```
    son(X,Y)              /вв/
    daughter(X,Y)         /вв/
    brother(X,Y)          /вв/
    grandmother(X,Y       /вв/
    grandfather (X,Y)     /вв/
    cusins(X,Y)           /вв/
    uncle(X,Y)            /вв/
    aunt(X,Y)             /вв/
```

Например:

```
parent(X,Y):- father(X,Y);mother(X,Y).     /вв/
husbend(X,Y):-wife(Y,X).                   /вв/
```

8. После ввода каждого нового предиката, программу следует откомпилировать, для чего нажать клавишу F9. Если при компиляции будут обнаружены синтаксические ошибки, то их следует исправить и добиться безошибочной компиляции (см. п. 2.2).

9. Если компиляция прошла успешно, перейти в режим исполнения: Run в главном меню или нажатие клавиш Alt-R.

10. В режиме исполнения каждый новый введенный вами предикат проверьте задавая в окне диалога после подсказки "Goal:" внешнюю цель с этим предикатом.

Например:

```
son(X,Y)   /вв/
```

после ввода соответствующего предиката.

11. Включить режим трассировки и просмотреть выполнение предиката по шагам (см. п. 2.3), нажимая клавишу F10 для выполнения каждого следующего шага. В окне трассировки будет показано связывание переменных и результат вычисления предиката.

12. Используя меню Setup, изменить размеры окон редактирования и трассировки.

13. По окончанию работы выйдите из системы выбрав Quit в меню Files или нажав клавиши Alt-X.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 2

Тема: Использование рекурсии.

Приведенная ниже база данных «путешествие» содержит факты, каждый из которых имеет по четыре аргумента. Каждый факт устанавливает, что можно совершить путешествие на транспортных средствах некоторой компании (аргумент 1) из одного города (аргумент 2 – пункт отправления) в другой город (аргумент 2 – пункт назначения) и при этом воспользоваться некоторым видом транспорта (аргумент 4).

```
%          компания    отправл.      прибытие      вид трансп.
путешествие(круизавиа,   благовещенск, москва,       самолет).
путешествие(автосервис,  благовещенск, белогорск,    автобус).
путешествие(ж/дсервис,   благовещенск, красноярск,   поезд).
путешествие(круизавиа,   москва,       сочи,         самолет).
путешествие(авиалинии,   красноярск,   брест,        самолет).
```

путешествие(ж/дсервис, тында, владивосток, поезд).

1. Правило «можно_путешествовать» устанавливает косвенную связь между городами, которая будет соблюдаться в том случае, если возможно путешествие из одного города в другой через третий – промежуточный – город. Декларативная трактовка данного правила может быть такой:

Будет возможным совершить путешествие между городами «Город А» и «Город Б», если можно добраться из города «Город А» в промежуточный пункт «Город Б» и можно добраться из города «Город Б» в «Город В».

или

путешествие из города «Город А» в «Город В» будет возможным, если либо 1) существует прямая транспортная связь между этими городами, либо 2) можно совершить путешествие из города «Город А» в некоторый промежуточный пункт «Город Б»Б а затем добраться из города «Город Б» в «Город В».

2. Правило «конкурент» гласит, что любые две транспортные компании будут конкурентами, если обе они обслуживают один и тот же маршрут. Декларативная трактовка данного правила может быть такой:

«Компания 1» будет конкурентом для компании «Компания 2», если существуют два города «Город А» и «Город Б», такие, что «Компания 1» обеспечивает перевозки между городами «Город А» и «Город Б», и «Компания 2» обеспечивает перевозки между городами «Город А» и «Город Б».

Последовательность действий

Изучить пояснения к работе.

Воспользуйтесь редактором для создания файла для работы с приведенными фактами и правилами.

Добавьте в программу 10 - 15 фактов для предиката "путешествие" и промоделируйте отношения, включая все необходимые факты и правила. Протестируйте Вашу программу, чтобы проверить, выполняет ли она то, что было задумано.

Составьте запросы, позволяющие:

a) получить сведения обо всех парах компаний - конкурентов;
b) получить сведения о возможности совершить путешествие из одного города в другой с выводом всех городов через которые прошел маршрут путешествия от исходного пункта к месту назначения.

Получить один ответ на запрос в следующем виде, например:

M=благовещенск --- самолет → москва --- поезд → смоленск

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 3

Тема: Работа со списками.

1. Создать случайным образом список состоящий из K нулей и единиц.
2. Написать предикат PL(L+,N-) – истинный тогда и только тогда, когда N – предпоследний элемент списка L, имеющего не менее двух элементов.
3. Определите возведение в целую степень через умножение и деление.
4. Вставить подсписок в определенное место списка.
5. Удалить все заданные элементы из списка.
6. Сложить два списка.
7. Напишите предикат subst(+V, +X, +Y, -L) – истинный тогда и только тогда, когда список L получается после взаимной замены X на Y, т.е. X → Y, Y → X.
8. Напишите предикат, который определяет, является ли данное натуральное число простым.
9. Напишите предикат p(+N, +K, -L) – истинный тогда и только тогда, когда L – список всех последовательностей (списков) длины K из чисел 1, 2, …, N.
10. Напишите предикат p(+N, -L) – истинный тогда и только тогда, когда список L содержит все последовательности (списки) из N нулей и единиц, в которых никакая цифра не повторяется три раза подряд (нет куска вида XXX).

ЛАБОРАТОРНАЯ РАБОТА № 2.

РАБОТА С ВНУТРЕННЕЙ И ВНЕШНЕЙ БАЗАМИ ДАННЫХ СИСТЕМЫ ТУРБО ПРОЛОГ

1. Концепция работы с файлами

Турбо Пролог поддерживает работы с файлами, а также работу с внутренней и внешней базами данных. Одновременно в системе можно производить операции с двумя файлами (устройствами): одним входным и одним выходным. Файлы объявляются в секции domains следующим образом:

```
domains
 file = name _file
```

Например:

```
domains
file = textfile    или   file = my_file
```

В системе имеются предопределенные файлы с именами: keyboard (клавиатура), printer (устройство печати), com1 (коммуникационный порт), screen (экран), stdin (стандартное устройство ввода), stdout (стандартное устройство вывода), stderr (стандартное устройство для сообщений об ошибках). По умолчанию stdin и stdout открыты и назначены соответственно на keyboad и screen.

Набор предикатов для ввода ориентирован на простые типы данных и включает в себя:

1) предикаты для работы с клавиатуры:

- inkey(Символ) (char): прототип (o) - читает символ со стандартного устройства ввода, если он доступен, иначе inkey считается невычисленным (fail).

Например:

```
clauses
  ready(X):-inkey(Y),X=Y,!.
```

При трассировке предикатов с inkey рекомендуется с помощью клавиш Alt-T установить Edit window в режим Off.

- keypressed - проверяет была ли нажата какая-либо клавиша на клавиатуре, если не была, то fail.

2) предикаты ввода данных:

- readln(Строка) (string), (o) - читает строку с текущего устройства ввода и связывает ее с заданной переменной.

Например:

```
readln(S)
```

- readterm(Область,Терм) (Name,Variable), (i,o) - читает объект, который был записан предикатом write. С помощью readterm осуществляется доступ к фактам в файле.

Например:

```
domains
   person=p(name,surname,height)
clauses
   readterm(person,p(N,S,H))
```

Файл, открытый при помощи предикатов openread и readdevice, содержит:

```
p("Seep","Maier",195)
```

Соответственно: N = Seep S = Maier H = 195.

- readint(Целочисленная_переменная) (integer), (o) - читает целое число с текущего устройства вывода и связывает его с заданной переменной.

- readreal(Переменная_вещественного_типа) (real), (o) - читает действительное число с текущего устройства ввода и связывает его с заданной переменной.

- readchar(Символьная_Переменная) (char), (o) - читает символ с текущего устройства ввода и связывает его с заданной переменной. В отличие от inkey устанавливает режим ожидания ввода.

- file_str(ИмяФайлаDOS,Строка) (string,string), (i,o) - читает строку из заданного файла и связывает ее с параметром "Строка". Максимально допустимый размер строки - 64 К. Признаком конца строки является символ Ctrl -Z (десятичный код ASCII = 26).

Например: file_str("B:TEXT1",X)

3) предикаты вывода данных:

- write(A1,A2,A3,…) (A1,A2,A3,… - константы или переменные), (i) - записывает заданные значения на текущее устройство вывода. Наряду с константами и переменными может использоваться также и обратный слэш. Он встречается в следующих комбинациях:

\n – выдается пробел;

\t – происходит переход к следующей позиции табуляции;

\номер – код ASCII выдаваемого символа.

Например: write("введенное имя",name) – на экране, который обычно является текущим устройством вывода, появляется текст: "введенное имя максим", если переменная name связана со строкой "максим"

- nl - выводит пустую строку

- writef(Формат, A1,A2,A3,…) (i) – предикат форматного вывода данных.

В строке формат после знака процента используются следующие ключи:

% g числа с плавающей точкой в наиболее компактном формате (по умолчанию).

% e числа с плавающей точкой в экспоненциальной форме.

% f числа с плавающей точкой в формате с фиксированной точкой.

Например: writef(″%5.2″,X) – результат 2.80, если X=2.8.

При работе с дисковым файлом его сначала следует открыть с указанием типа выполняемых затем действий: чтения, записи, добавления текста в конец файла или модификации содержимого файла. Это делается с помощью следующих предикатов:

- openread(Символическое_имя_файла, Имя_файла_в_DOS) (file, string), прототип (i,i) – открывает файл для чтения.

Например: openread(td,″C:text.dat″) – файл DOS text.dat на устройстве C будет открыт для чтения под именем td.

- openwrite (Символическое_имя_файла, Имя_файла_в_DOS) (file, string), прототип (i,i) – открывает файл для записи.

Например: openwrite(users,″B:us.dat″) – файл DOS us.dat на устройстве B будет открыт для записи под именем users.

- openappend(Символическое_имя_файла, Имя_файла_в_DOS) (file, string), прототип (i,i) – открывает файл для дополнения.

Например: openappend(persons,″B:person.txt″) – файл DOS person.txt на устройстве B будет открыт для дополнения под именем persons.

- openmodify(Символическое_имя_файла, Имя_файла_в_DOS) (file, string), прототип (i,i) – открывает файл для модификации (как для чтения, так и для записи).

Например: openmodify(addr,″A:ADDRESSES″) – файл DOS ADDRESSES на устройстве A будет открыт для чтения или записи под именем addr.

Если в предикате openwrite задано имя файла уже существующего на диске, то после выполнения openwrite содержимое этого файла будет стерто. Проверить наличие на диске указанного файла, особенно если файл открывается не для записи, можно до выполнения операции его открытия с помощью предиката

- existfile(Имя_файла_в_DOS) (string), прототип (i) – выполняется успешно, если заданный файл присутствует в текущем каталоге, и завершается неудачно в противном случае.

Одновременно может быть открыто несколько файлов, но только два их них могут являться текущими файлами (устройствами) ввода и вывода, которые объявляются предикатами:

- readdevice(Имя_файла) (symbol): прототип (i) – присваивает текущему устройству вводимое заданное символическое имя файла; протопит (o) – связывает с параметром ″Имя_файла″ символическое имя файла текущего устройства ввода.

Например:

readdevice(adr) – последующие команды чтения осуществляют чтение из файла *adr*.
readdevice(X) – результат X=keyboard.

- writedevice(Символическое_имя_файла) (symbol): прототип (i) - ставит в соответствие текущему устройству вывода заданное символическое имя файла; протопит (o) – связывает с параметром имя текущего устройства вывода.

Например:

writedevice(addresses) – последующие команды записи могут использовать символическое имя файла *addresses;*
writedevice(X) – результат X= *addresses*.

По умолчанию файлы считаются текстовыми, однако установив режим работы с файлом можно работать с ним как с двоичным, однако при этом следует использовать только посимвольный ввод.

- filemode(Символическое_имя_файла, Тип Файла) (file,integer): прототип (i,i) режим: 0 – текстовый файл, 1 –двоичный файл – устанавливает тип заданного файла; прототип (i,o) – читает тип заданного файла и связывает его с параметром ″Тип файла″.

Например:

filemode(users,0) – тип файла *users* устанавливается как текстовый;
filemode(users,X) – результат X=0, если файл *users* – текстовый.

По завершению работы с файлом его следует закрыть:

- closefile(Символическое_имя_файла) (file): прототип (i) – имя файла не должно быть заключено в кавычки. Выполняется успешно, даже если файл перед этим не был открыт.

Например:

```
domains
        file = textfile
goal
openread(textfile, "goo.txt"), readdevice(textfile),
readln(Str),write(Str),closefile(textfile).
```

Файлы на диске из программы могут быть переименованы или удалены:

- renamefile(Старое_DOS_имя, Новое_DOS_имя) (string,string), (i,i)
- deletefile(Имя_файла_в_DOS) (string),(i).

К вспомогательным предикатам относятся:

- filepos(Символическое_имя_файла, Позиция_в_файле, Режим) (file,real,integer) - устанавливает указатель данного файла на заданную позицию. Режим 0 = относительно начала файла;
  1 = относительно текущего положения указателя;
  2 = относительно конца файла.

Например:

filepos(abc,10,0) – устанавливает указатель в файле *abc* на десятом байте.

- eof(Символическое_имя_файла) (file),(i) – выполняется успешно, если указатель текущей позиции файла указывает на конец файла, и завершается неудачно в противном случае.

- flush(Символическое_имя_файла) (file), (i) – содержимое внутреннего файлового буфера пересылается в заданный файл.

- disk(DosPath) /* (string): прототип (i) – устанавливает путь и накопитель; (o) – связывает с параметром текущий накопитель и путь.

Например:

disk(″C:\Prolog″) – на накопителе C будет установлен путь \Prolog.
disk(X) – результат X = C:\Prolog\Bin\qwer.

2. Внутренняя база данных

Турбо Пролог поддерживает работу с внутренней (ВБД) и внешней (дисковой - ДБД) базами данных, а также работу с дополнительной оперативной памятью (EMS ОЗУ).

ВБД состоит из фактов, которые добавляются/удаляются и существуют в ОЗУ только во время работы программы. Однако, их можно сохранить на диске. Для работы с ВБД Турбо Пролог использует следующие встроенные предикаты [1]:

- assert(факт) – заносит факт в базу данных перед другими фактами. В результате данный факт будет добавлен в начало базы данных. Факт должен быть термом, принадлежащим домену *dbasedom.*
- asserta(факт) - добавляет факты в ВБД перед существующими фактами.
- assertz(факт) - добавляет факты в ВБД после существующих фактов
- retract(факт) - удаление существующего факта из ВБД
- retractall(факт) - удаление существующих фактов из ВБД
- consult(имя_файла) – записывает в базу данных текстовый файл, который может быть создан в результате выполнения предиката *save*. Этот файл содержит факты, которые должны быть описаны в разделе *database*. Выполнение предиката *consult* не будет успешным, если в файле имеются синтаксические ошибки.
- save(имя_файла) - сохранение ВБД в файле или на жесткий диск.

Эти предикаты могут иметь один или два аргумента. Второй аргумент является необязательным и обозначает имя ВБД. Факты в ВБД хранятся в виде таблицы, которую легко модифицировать. В ВБД можно добавлять факты, но не правила. В фактах ВБД не может быть свободных переменных. Если ВБД не дано имя, то ей будет присвоено имя dbasedom.dba.

Пример программы с использованием внутренней базы данных:

```
domains
database
    base(integer,string,integer)
predicates
    write_base(integer)
    ch_base(integer)
clauses
     write_base(N):-base(N,Name,Group),
                    write("Номер ",N),nl,
                    write("Фамилия ",Name),nl,
                    write("Группа ",Group),nl,
                    write("----------------"),nl.
      ch_base(N):-write("Фамилия:"),readln(S),
                  write("Группа"),readint(S1),
                  retract(base(N,_,_)),
                  assert(base(N,S,S1)).
   goal
       consult("dd.dba"),
       write("Изменить запись.Номер-"),
       readint(N),
       write_base(N),
       ch_base(N),
       save("dd.dba").
```

3. Внешняя база данных

Внешняя база данных создается в случае, если объем данных больше объема свободной части ОЗУ или предполагается значительное расширение БД.

ДБД может быть расположена:

Встроенный атрибут	Местонахождение внешней БД

In_file	в файле на диске
In_memory	в оперативной памяти
In_ems	в EMS-памяти (расширение ОЗУ до 4 Мб.)

ДБД состоит из двух компонент: элементов данных (термы Турбо Пролога) и цепочек (chain), в которых хранятся термы. В цепочке может храниться неограниченное количество термов. ВДБД может быть любое число цепочек. Цепочка выбирается по имени. Имя цепочки – это просто строка символов. Для быстрого поиска данных цепочка может быть индексирована методом B+дерева [1].

Имена предикатов, работающих с ДБД, построены следующим образом: db_ <тип объекта данных><операция>.

Например: db_term_delete.

Если имя объекта не указано подразумевается вся база данных. Объявление имени ДБД:

```
domains
  db_selector = имя_базы1, имя_базы2, ...
```

Ссылка на ДБД осуществляется по ее имени.

3. 1. Предикаты для работы со всей ДБД.

- db_create(имя_базы, имя_файла, расположение), (i,i,i) – создание новой ДБД:

Например:

```
db_create(db_name, "F.TXT", in_file)
db_create(db_mark, "marks", in_memory)
```

- db_open(имя_базы, имя_файла, расположение), (i,i,i) – открытие ранее созданной базы данных.

- db_copy(Имя_базы, имя_файла, Новое_расположение), (i,i,i) – перемещение бызы данных в другое место (например, из файла в ОЗУ или наоборот.

- db_close(Имя_базы), (i) – в конце работы ДБД должна быть закрыта.

- db_delete(Имя_базы), (i) – ДБД можно удалить.

3.2. Предикаты для работы с цепочками

Цепочки [3] ДБД аналогичны внутренней базе данных. Цепочки, в некотором смысле, можно отожествить с записями в обычной реляционной СУБД. Они хранят данные в форме термов Пролога (т.е. в виде сложных объектов: списков, структур; или значений простых типов данных). ДБД может хранить любые допустимые в Турбо – Прологе типы термов.

Цепочки термов запоминаются последовательно и одновременно доступна только одна цепочка. Каждая цепочка имеет собственное имя. Имя цепочки нигде специально не объявляется, цепочка создается, когда ее имя впервые встретилось в предикатах записи фактов в ДБД. Поэтому в именах цепочек не следует допускать опечаток, иначе можно создать ненужную цепочку и таким образом "потерять данные".

Следующие предиката служат для работы с цепочками в ДБД:

- chain_inserta(X,W,S,T,_) – вставляет терм в начало базы;
- chain_insertz(X,W,S,T,_) – вставляет терм в конец базы;
- db_chains(X,C) – создает ссылку на цепь;
- chain_first(X,C,R) – позиционирует указатель на первый элемент цепи;
- ref_term(X,S,R,TERM) – возвращает терм на который указывает указатель;
- chain_next(X,R,NEXT) – возвращает указатель на следующий терм,

где X – селектор базы описывается в DOMAINS как DB_SELECTOR = mydba

Y – имя базы по стандарту ДОС в кавычках

Z – место расположения базы (в памяти или в виде файла)

W – описатель цепи

S – описатель структуры файла базы данных

T – терм

_ – необъязательная переменная возвращающая код ошибки если она есть

C – ссылка на описатель цепи

R – положение указателя в цепи

TERM – переменная конкретизированная термом на котором находится указатель

NEXT – переменная конкретизированная указателем на следующий терм

dbman – описатель структуры файла базы данных

4. Индексация цепочек методом B+дерева.

При добавлении термов с использованием для индексации B+дерева с каждым ссылочным числом связан ключ. Так как это число ссылается на уникальную запись (вход) БД, нахождение правильного ключа быстро приводит к искомой записи данных. B+дерево (индексная цепочка) создается с помощью предиката:

- bt_create(Имя_ДБД, Имя_B+дерева, Селектор_B+дерева, Длина_Ключа, Длина_Узла), так как информация B+дерева хранится в том же файле ДБД, что и индексируемая цепочка, то необходимо указывать "Имя_ДБД" базы, в которой находится эта цепочка. "Имя_B+дерева" - это имя, которое ему дает пользователь; оно является строкой. Третий аргумент является выходным, он относится к специальному встроенному простому типу bt_selector и используется в ряде других предикатов для идентификации созданного B+дерева. "Длина ключа" - длина самого длинного ключа по которому осуществляется поиск записи. Однако не следует злоупотреблять этим аргументом, так как хранение слишком длинных ключей в большой ДБД может потребовать значительного объема дисковой памяти.

B+дерево разбито на отдельные страницы или узлы. Каждый узел B+дерева является либо последним, либо порождает два нижележащих узла. Каждый узел содержит группу ключей из заданного диапазона. Так как никакие два узла не содержат одинаковые значения ключей, то можно быстро проверить не находится ли искомый ключ внутри диапазона конкретного узла. Если да, то быстро находится ссылочное число, если ключ меньше, то поиск ведется по левой ветви B+дерева, если больше - по правой. Аргумент "Длина_Узла" в bt_create определяет сколько ключей запоминается в каждом узле B+дерева. Для баз данных среднего размера рекомендуется "Длина_Узла" = 4.

Как и цепочки ДБД, B+деревья могут быть закрыты и снова открыты с помощью предикатов: bt_open() и bt_close().

Чтобы B+дерево было правильно сохранено, оно должно закрываться до закрытия соответствующего файла базы данных. Как и с ДБД B+дерево может быть модифицировано удалением или добавлением ключей. При индексировании ключи автоматически сохраняются системой. Для модификации используются два предиката: key_insert() и key_delete()

Для поиска по заданному ссылочному номеру следует использовать предикат key_search(), если заданный ключ не найден, возникает ситуация fail. Если существует 2 одинаковых ключа, то возвращается первый найденный. Чтобы найти следующий (предыдущий) нужно применить, соответственно, предикат
key_next() или key_prev(). Полный список предикатов для работы с внитренней и внешней базами данных можно получить в процессе работы нажав клавишу F1. Они находятся в файле PROLOG.HLP.

Примеры программ для работы с ДБД:

1)
```
domains
    db_selector = mmm
predicates
    count_dba(integer)
    count(REF,INTEGER,INTEGER)
clauses
    count_dba(N):- chain_first(mmm,name,REF),count(REF,1,N).
    count(REF,N,N):-ref_term(mmm,string,REF,S),write(S).
    count(REF,N,N2):-chain_next(mmm,REF,NEXT),!,N1=N+1,
    count(NEXT,N1,N2).
```

```
goal
    db_create(mmm, "aaaa.ile", in_file),
    db_statistics(mmm,_,_,_,_),
    write("\nБаза имен.\n"),
    write("Введите номер не больше 8"),
    readint(N),
    chain_inserta(mmm,name,string,"ДМИТРИЙ",_),
    chain_insertz(mmm,name,string,"АЛЕКСЕЙ",_),
    chain_insertz(mmm,name,string,"МАКСИМ",_),
    count_dba(N),
    db_close(mmm).
```

2)
```
domains
        db_selector = mydba
        dbman = person(firstname,age,city)
        firstname,city = srting
        age = integer
    predicates
        rd(ref)
    clauses
        rd(REF):-ref_term(mydba,dbman,REF,TERM),write(TERM),nl,fail.
        rd(REF):-chain_next(mydba,REF,NEXT),!,rd(NEXT).
        rd(_).
    goal
        clearwindow,
        db_create(mydba,"dd.bin",in_file),
        db_close(mydba),
        db_open(mydba,"dd.bin",in_file),
chain_inserta(mydba,mychain,dbman,person
                ("Унру Артур Яковлевич",21,"Комсомольск-на-Амуре"),_),
chain_insertz(mydba,mychain,dbman,person
                ("Смирнов Николай Семенович",20,"Благовещенск"),_),
        db_chains(mydba,CHAIN),
        chain_first(mydba,CHAIN,REF),
        rd(REF),readchar(_),
        db_delete("dd.bin",in_file).
```

Последовательность действий

1. Изучить пояснения к работе.
2. Загрузить Турбо Пролог.
3. Войти в режим редактирования (Alt-E) и ввести одну из предложенных в примере программ.
4. Включить режим трассировки и просмотреть выполнение предикатов по шагам (см. п. 2.3 предыдущей работы).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 1

Тема: Работа с базами данных.

1. Разработать программу с использованием файлов, ВБД, ДБД по темам:

1.1. Самолеты: наименование типа, фамилия конструктора, год выпуска, количество кресел, грузоподъемность (т).

1.2. Расчет движения: наименование воздушной линии, тип самолета, количество рейсов, налет (тыс. км.), пассажирооборот.

1.3. Перевозки: тип самолета, номер борта, количество рейсов, налет в часах, налет (тыс. км.).

1.4. Расписание: номер рейса, наименование рейса, тип самолета, стоимость билета, протяженность линии.

1.5. Сооружения аэропорта: наименование, площадь, этажность, год сооружения, стоимость (млн. руб.).

1.6. Ремонт аэродромных сооружений: наименование, шифр, вид ремонта, стоимость ремонта, наименование подрядчика.

1.7. Кассы авиабилетов: номер кассы, Ф.И.О. кассира, количество проданных билетов, суммарная выручка, дата продажи.

1.8. Характеристики персональных компьютеров: тип процессора, тактовая частота, емкость ОП (Мбайт), емкость ЖМД (Мбайт), тип монитора.

1.9. Города: наименование, количество жителей, площадь (кв. км.), год основания, количество ВУЗов).

1.10. Московские мосты: наименование, высота, ширина, количество опор, протяженность.

1.11. Линии московского метро: наименование, район линии, год пуска, протяженность (км.), количество поездов.

1.12. Легковые автомобили: марка, цвет, стоимость, изготовитель, максимальная скорость.

1.13. Продажа программных продуктов: наименование, фирма – изготовитель, стоимость (тыс. руб.), объем (Мбайт), количество на складе.

1.14. Абонентская плата за телефон: Ф.И.О. абонента, телефон, год установки, количество абонентов, плата за телефон.

1.15. Детские сады: наименование детского сада, номер сада, количество детей, район города, плата за месяц.

1.16. Сотрудники: Ф.И.О., табельный номер, дата рождения, оклад (тыс. руб.), стаж работы.

1.17. Ведомость зарплаты за текущий месяц: Ф.И.О., номер отдела, Табельный номер, количество рабочих часов, размер зарплаты.

1.18. Музеи: наименование, назначение, адрес, время работы, стоимость билета.

1.19. Экскурсии: наименование, страна, стоимость, продолжительность, транспорт.

1.20. Кинофильмы: наименование кинотеатра, стоимость билета, время сеансов, адрес мест, количество.

1.21. Книга – почтой: наименование, Ф.И.О. автора, номер по каталогу, издательство, стоимость книги.

1.22. Квартиры: адрес, площадь (кв. м.), сторона света, стоимость (1 кв. м.), этаж, количество комнат.

1.23. Склад товаров: номер магазина, наименование товара, артикул товара, цена единицы товара, количество товара.

1.24. Телевизоры на складе магазина: наименование, фирма – изготовитель, стоимость, размер экрана, количество на складе.

1.25. Холодильники на складе магазина: наименование, фирма – изготовитель, стоимость, емкость камеры, количество на складе.

2. Пользуясь средствами Турбо Пролога добавить предикаты для поиска фактов в построенной внутренней базе данных.

3.Каждый новый введенный вами предикат проверить задавая в окне диалога после подсказки Goal: внешнюю цель с этим предикатом.

ЛАБОРАТОРНАЯ РАБОТА № 3.
УНИВЕРСАЛЬНЫЙ ГРАФИЧЕСКИЙ ИНТЕРФЕЙС В ЯЗЫКЕ *ТУРБО ПРОЛОГ.*

Общие сведения

1. Режимы работы монитора

1.1. Определения

Текстовый режим [4] определяет вывод на экран, который разделен на ячейки (обычно 80 колонок на 25 строк), и может отображать только символы из кодового набора ПЭВМ. Каждая ячейка содержит атрибут и символ. Атрибут говорит о том, как выводится на экран символ (его цвет, мигание, интенсивность свечения).

Текущая позиция на экране отмечается мерцающим прямоугольником - курсором. Позиция курсора X,Y (координаты точки) задаются номером колонки и номером строки символа или точки в зависимости от режима. Началом координат является левый верхний угол экрана (1,1), при этом X увеличивается слева направо, а Y – сверху вниз.

Окно – прямоугольная область экрана (или весь экран), возможно окруженная рамкой и выделенная другим цветом или оттенком. Операции над окном производятся как над целым экраном. На экране одновременно могут находиться несколько окон, которые могут перекрывать друг друга. Окно, в котором находится курсор, называется текущим.

Для кодирования цвета окон в предикатах Турбо – Пролога приняты следующие соглашения [2]:

Цвет фона	Код	Цвет изображения	Код
черный	0	черный	0
голубой	16	голубой	1
зеленый	32	зеленый	2
сиреневый	48	сиреневый	3
красный	64	красный	4
фиолетовый	80	фиолетовый	5
коричневый	96	коричневый	6
белый	112	белый	7

Для получения яркого цвета к цвету изображения нужно добавить 8. Для получения атрибута окна нужно сложить цвет фона и изображения. Например, голубое изображение на белом фоне имеет атрибут: 112+1=113.

Графический режим [4] определяет построение изображения на экране из точек (пикселов). Размер экрана измеряется количеством точек по горизонтали (X) и по вертикали (Y) и зависит от типа видеоадаптера. Число точек по осям X,Y называется разрешением экрана. Каждая точка может быть включена или выключена, а на цветных мониторах иметь еще и цвет. Таким образом изображение на экране строится включением и окраской точек.

В графическом режиме нет курсора. Вместо него используется указатель текущей позиции экрана CP, который может быть перемещен в любое место экрана, где должен строиться элемент изображения.

При наличии видеоадаптера цветной графики (CGA, MCGA, VGA) можно установить цвет фона (экрана) и цвет изображения. Цвет фона и цвет изображения являются атрибутом окна и изображения. В процессе работы атрибут можно менять.

Турбо Пролог 2.0 поддерживает следующие видеоадаптеры [1]:

Название видеоадаптера	Драйвер
CGA - Color Graphics Adapter	CGA.BGI
EGA - Enhanced Graphics Adapter	EGAVGA.BGI
MCGA - Multi Color Graphics Adapter	
HGA - Hercules Graphics Adapter	HERC.BGI
VGA - Video Graphics Array	EGAVGA.BGI
IBM8514 - Super VGA	IBM8514.BGI
AT&T - 400-строковый, ПЭВМ Оливетти	ATT.BGI

Основными видеоадаптерами являются: CGA, EGA и VGA.

Графика CGA.

Графический адаптер CGA был первым цветным графическим адаптером на ПЭВМ фирмы IBM. CGA поддерживает один текстовый режим (25*80) и два отдельных графических режима: с высоким (640*200) инизким (320*200) разрешением экрана. При

низком разрешении палитра (набор цветов) одновременно выводимых на экран цветов состоит из 4 цветов, один из которых является цветом фона.

В режиме с высоким разрешением можно использовать только один цвет изображения на черном фоне.

Графика EGA.

EGA значительно расширяет графические возможности ПЭВМ. Как и CGA, видеоадаптер EGA может быть инициализирован для работы в двух режимах: низкого (640*200) и высокого (640*350) разрешения экрана. Оба эти режима позволяют выводить на экран одновременно палитру из 16 различных цветов.

Главным различием между этими режимами является количество предоставляемых буферных страниц. В режиме низкого разрешения в памяти платы EGA может быть сохранено четыре полных страницы графического экрана. Это позволяет пользовательской программе создавать графическое изображение в трех различных экранах во время вывода на экран четвертого. Эти три дополнительных экрана дают возможность строить изображения на невидимых экранах, затем быстро переходить от одного графического экрана к другому не ожидая, пока будет нарисовано новое изображение.

В режиме высокого разрешения EGA доступны две страницы: одна не-видимая и одна выведенная на экран.

Графика VGA.

Имеется совместимость с EGA и CGA. Максимальная палитра 256 цветов. Разрешение 640*480. Окна в графическом режиме. Как и окна текстового режима, окно в графическом режиме, именуемое в дальнейшем VP (viewport), используются для выбора конкретной части экрана для построения в нем изображения. По умолчанию VP после инициализации занимает весь экран. В отличие от окон текстового режима VP не буферизируются. Это означает, что при перекрытии окон, информация в первом из них будет потеряна.

При записи предикатов в общем виде в строке комментарий указывается тип аргумента и вид передачи аргументов (текущий шаблон), т.е. какие аргументы являются входными (i), а какие выходными (o). Некоторые предикаты могут иметь несколько различных текущих шаблонов в зависимости от того, получают они связанные или несвязанные переменные.

1.2. Работа с окнами

Основным предикатом для создания окон является предикат для определения окна:

- makewindow(Ном_Окна, Атр_экрана, Атр_рамки, Заголовок, Строка, Столбец, Высота, Ширина,), где

Ном_Окна – логический номер, присваиваемый окну для ссылок в других предикатах. Номера 80-85 использовать не рекомендуется, так как они зарезервированы за пакетом TOOLBOX (см. раздел 3.);

Атр_экрана – атрибут цвета для внутренней области окна;

Атр_рамки – атрибут цвета для внешней области окна (рамки), 0 – если рамка отсутствует;

Заголовок – текст, помещаемый на верхней рамке окна;

Строка – номер строки верхнего левого угла окна;

Столбец – номер колонки верхнего левого угла окна;

Высота – высота окна в строках;

Ширина – ширина окна в колонках.

Например:

makewindow(1,7,135,"Пример окна",5,15,6,10)

Параметры должны соответствовать характеристикам аппаратуры, иначе выдается сообщение об ошибке. Созданное окно становится текущим окном. Экран может быть разбит на несколько окон.

Для перехода их одного окна в другое служит предикат

- shiftwindow(Ном_Окна): прототип (i) – активизирует окно с заданным номером; прототип (o) – связывает с параметром номер текущего окна.

Например:

shiftwindow(3) – активизируется окно с номером 3;

shiftwindow(X) – результат: X=1, если текущее окно имеет номер 1.

При переходе в другое окно координаты являются локальными т.е. отсчет идет от левого верхнего угла, которое имеет координаты (0,0).

- clearwindow – стирает содержимое текущего окна (очищает окно), при этом курсор перемещается в позицию (0,0);

- removewindow – удаляет текущее окно, если окна не существует, выдается ошибка времени выполнения.

Управление курсором осуществляется с помощью предиката

- cursor(Строка, Столбец): прототип (i,i) – передвигает курсор в текущем окне в позицию, заданную номерами строки и столбца. Номер строки может находится в диапазоне от 0 до 24, номер столбца – от 0 до 79. координаты левого верхнего угла экрана – (0,0); прототип (o,o) – связывает номера строки и столбца, определяющие позицию курсора, с соответствующими параметрами.

Например:

cursor(5,10) – курсор устанавливается на пятой строке в десятом столбце;

cursor(Z,S) – результат: Z=12, S=40, если курсор установлен на строке 12 в столбце 40.

- existwindow(Ном_Окна) – проверяет наличие окна с заданным логическим номером;

- setcolor(Окно_или_рамка) – позволяет пользователю интерактивно изменять цвет окна (0) или рамки (1). При этом появляется меню, аналогичное подменю Setup/Colors в интегрированной оболочке Турбо Пролога.

- attribute(Атрибут): прототип (i) – устанавливает значение атрибута, определяющее цвет фона текущего окна. Значение атрибута определяется параметром (см. табл.); прототип (o) – опрашивает атрибут текущего окна и связвает его с параметром.

Например:

attribute(88) – устанавливает розовый цвет текущего окна;

attribute(X) – если фон окна голубой, то значением переменной X будет число 24.

- scrol (Число_строк, Число_столбцов): прототип (i,i) – сдвигает содержимое текущего окна на заданное число строк и столбцов. Положительные значения аргументов задают скроллинг вниз и вправо, а отрицательные - вверх и влево.

Пример программы для работы с окнами:

```
predicates
        nondeterm repeat
goal
        makewindow(1, 1, 7, "one", 5, 0, 10, 20), write("ONE"),
        makewindow(8, 8, 7, "eight", 1, 10, 10, 20), write("EIGHT"),
        makewindow(9, 9, 7, "nine", 15, 20, 10, 20), write("NINE"),
        repeat,
        random(9,X), N=X+1, shiftwindow(9),
         shiftwindow(1),
         keypressed.
clauses
         repeat.
         repeat :- repeat.
```

2. Универсальный графический интерфейс (УГИ)

УГИ – это свыше 70 встроенных предикатов для работы с графикой. Эти предикаты позволяют работать как с отдельной точкой, так и с объектами типа линий, дуг, окружностей, многоугольников и др.

УГИ поддерживает различные виды штриховки линий, закраску поверхностей, различные виды шрифтов (фонты) и драйверы практически всех распространенных графических адаптеров ПЭВМ.

Полный список предикатов УГИ можно получить в процессе работы нажав клавишу F1. Они находятся в текстовом файле PROLOG.HLP. Чтобы применять УГИ необходимо иметь видеографический адаптер, поддерживаемый УГИ (например, нельзя применять на ЕС 1840).

2.1. Установка графического режима

Режим работы графического экрана устанавливается предикатом initgraph, который инициализирует графическую систему, загружая с диска соответствующий драйвер графического видеоустройства (см. файлы с расширением BGI) переводя его в графический режим работы. Предикат initgraph() имеет пять аргументов. Он связывает переменные NewDriver и NewMode с реальным драйвером и режимом работы, а также устанавливает значения по умолчанию всем графическим переменным (текущая позиция, цвет, палитра и т.д. в зависимости от видеоадаптера).

- initgraph(Graphdriver,Graphmode,NewDriver,NewMode,Pathtodriver) (integer,integer,integer,integer,string): прототип (i,i,o,o,i), где

Graphdriver – целое число, которое задает номер используемого драйвера, 0 означает, что система должна сама определить тип используемого в ПЭВМ видеоконтроллера и вернуть номер соответствующего ему драйвера (1 – CGA, 2 – MCGA, 3 – EGA, ..., 9 – VGA). См. также декларации в файле GRAPDECL.PRO;

Graphmode – графический режим. Символические константы для объявления режима определены в файле GRAPDECL.PRO.

Pathtodriver – путь к директории, где находится драйверы (*.BGI), пустая строка " " означает текущую директорию.

Например:

```
constants
  bgi_path = " "
goal
  InitGraph(G_Driver,G_Mode, _, _, bgi_Path)
```

После инициализации экрана он очищается и СР устанавливается равным координатам верхнего левого угла экрана (0,0). Иногда, если предполагается, что программа будет выполняться на различных ПЭВМ, более удобно перед выполнением предиката initgraph() выполнить предикат:

- detectgraph(Graphdriver,Graphmode) (integer,integer): прототип (o,o) – проверяет какой графический адаптер используется в системе и определяет режим его работы при наибольшем разрешении. Если графический адаптер не обнаружен, то возвращается – 2.

Например:

```
DetectGraph(G_Driver, G_Mode1),
```

Чтобы освободить память, занятую под графику, и вернуть экран в режим, используемый до initgraph, в конце работы нужно выполнить предикат: closegraph().

2.2. Перемещение по экрану

Так как характеристики видеотерминалов значительно отличаются, то с помощью следующих предикатов рекомендуется получить максимальные значения X и Y: getmaxX(X) и getmaxY(Y).

Для перемещения по плоскости изображения используются предикаты:

- getx(X) – получить текущую координату X;
- gety(Y) – получить текущую координату Y;
- moveto(X,Y) – переместить текущий указатель в точку X,Y;
- moverel(DeltaX,DeltaY) – переместить текущий указатель на DeltaX пикселов по горизонтали и на DeltaY пикселов по вертикали.

Предикат cleardevice не имеет аргументов. Он очищает графический экран перемещает указатель текущей позиции в точку (0,0).

2.3. Установка/изменение текущих значений

Предикаты установки значений:

- setlinestyle((Вид_Линии, Шаблон, Толщина) (integer,integer,integer): прототип (i,i,i) – устанавливает вид линии по умолчанию, используемый другими предикатами УГИ. Параметры: "Вид_Линии" – сплошная (0), из точек (1), центрированная (2), пунктирная(3), задаваемая пользователем (4); "Шаблон" – 16-битовый шаблон задающий вид линии (только для "Вида_Линии"=4, иначе шаблон игнорируется): сплошная линия $FFFF, пунктирная $3333; "Толщина" – нормальная линия или утолщенная.

- setfillpattern(Список_шаблона, Цвет) (bgi_list,integer): прототип (i,i) – устанавливает определяемый пользователем шаблон – заполнитель. Шаблон в виде матрицы 8 на 8 бит кодируется в виде списка из 8 однобайтовых элементов, каждый из которых кодирует 8 бит. Если бит шаблона равен 1, то соответствующий ему пиксел будет выведен на экран.

- setfillstyle(Шаблон, Индекс_Цвета) (integer,integer): прототип (i,i) – устанавливает в качестве текущего заполнителя один из определенных в системе шаблонов, а текущий цвет заполнителя равным "Индекс_Цвета" в палитре. Константы для "Шаблона" приведены в файле GRAPDECL.PRO

- setpalette(Индекс, Реальный_Цвет) (integer,integer): прототип (i,i) – изменяет один цвет палитры, расположенный в ней под номером "Индекс", на цвет заданный параметром "Реальный_Цвет". Последний представляет собой зависящий от аппаратуры номер цвета для используемого драйвера. Индекс должен находится в диапазоне от 0 до <количество_цветов_в_текущей_палитре>. Для адаптера CGA можно изменять цвет только с индексом 0 (цвет фона). Файл GRAPDECL.PRO содержит определения констант для наиболее часто применяемых цветов.

- setallpalette(Список_Цветов) (bgi_ilist): прототип (i) – изменяет все цвета палитры в соответствии со "Списком_Цветов" (адаптеры EGA/VGA).

Текущий цвет фона и цвет изображения (из текущей палитры) можно получить/установить с помощью предикатов:

- getbkcolor(BkColor) (integer): прототип (o) – возвращает цвет фона;
- getcolor(Color) (integer): прототип (o) – возвращает цвет изображения;
- setbkcolor(Color) (integer): прототип (i) – установить цвет фона;
- setcolor(Color) (integer): прототип (i) – установить цвет изображения.

Следующие два предиката позволяют получить/изменить цвет заданной точки:

- getpixel(X,Y,Цвет) (integer,integer,integer): прототип (i,i,o);
- putpixel(X,Y,Цвет_Точки) (integer,integer,integer): прототип (i,i,i);

где (X,Y) - координаты точки (пиксела).

2.4. Построение графических объектов

В следующих предикатах углы отсчитываются против часовой стрелки, 0 градусов соответствует трем часам.

- arc(X,Y,StartAngle,EndAngle,R) (integer,integer,integer,integer,integer): прототип (i,i,i,i,i) – рисует текущим цветом дугу окружности с центром (X,Y) и радиусом R, начинающуюся с ″StartAngl″ и заканчивающуюся на ″EndAngl″;

- ellipse(X, Y, StAngle, EndAngle, Xradius, YRadius) (все аргументы целочисленные): прототип (i,i,i,i,i,i) – рисует эллипс;

- pieslice(X,Y,Нач_Угол, Кон_Угол, Радиус) (все аргументы целочисленные): прототип (i,i,i,i,i) – рисует и заполняет сектор круга от ″Нач_Угол″ до ″Кон_Угол″ с центром в (X,Y) и заданным радиусом. Граница сектора выделяется текущим цветом изображения, а заполнение производится текущим шаблоном и цветом заполнителя;

- pieslicexy(X,Y,Нач_Угол, Кон_Угол, X_Радиус, Y_Радиус) (все аргументы целочисленные): прототип (i,i,i,i,i,i) – рисует и заполняет сектор эллипса с центром (X,Y),

горизонтальным радиусом ″X_Радиус″, вертикальным – ″Y_Радиус″, от ″Нач_Угол ″до ″Кон_Угол″. Остальное как для pieslice;

- line(X0,Y0,X1,Y1) (все аргументы целочисленные): прототип (i,i,i,i) – рисует текущим цветом, стилем линии и толщиной прямую линию между двумя заданными точками: (X0,Y0) – начало; (X1,Y1) – конец. Предикат line не обновляет текущую позицию и всегда вычисляется;

- linerel(DeltaX,DeltaY) (integer,integer): прототип (i,i) – рисует линию от текущей позиции до точки приращения координат до которой равны (DeltaX, DeltaY). Текушая позиция изменяется на (DeltaX, DeltaY);

- lineto(X,Y) (integer,integer): прототип (i,i) – рисует линию от текущей позиции до точки с координатами (X,Y), после чего текущая позиция перемещается в (X,Y);

- rectangle(X, Y, X1, Y1) (все аргументы целочисленные): прототип (i,i,i,i) –рисует текущим цветом и стилем линии прямоугольник по координатам его верхнего левого (X,Y) и нижнего правого (X1,Y1) углов;

- drawpoly(PolyPointsList) (point_list): прототип (i) – рисует многоугольник,
где (point_list) – список координат вершин X,Y.

Например: drawpoly([45,15, 85,45, 45,85, 15,45]).

- circle(X,Y,Radius) (integer,integer,integer): прототип (i,i,i) – рисует окружность.

При построении окружности учитывается отношение масштабов по горизонтали и по вертикали.

- getaspectratio(Xasp,Yasp) (integer,integer): прототип (i,i) – вычисляет коэффициенты искажения по горизонтали Xasp и по вертикали Yasp; прототип (o,o) – устанавливает новые коэффициенты;

- bar(Left,Top,Right,Bottom) (все аргументы целочисленные): прототип (i,i,i,i) – рисует заполненную текущим шаблоном прямоугольную полосу;

- bar3d(Left,Top,Right,Bottom,Depth,Topflag) (все аргументы целочисленные): прототип (i,i,i,i,i,i) – рисует заполненную текущим шаблоном трехмерную прямоугольную полосу;

- outtext(СтрокаТекста) (string): прототип (i) – выводит строку текста в окно вывода (viewport) используя текущий шрифт, размер, направление и установки выравнивания;

- outtextxy(X,Y, СтрокаТекста) (integer, integer, string): прототип (i,i,i) – выводит с заданной позиции (X,Y) строку текста в окно вывода используя текущий шрифт, размер, направление и установки выравнивания;

- getimage(Left,Top,Right,Bottom,BitMap) (все аргументы целочисленные): прототип (i,i,i,i,o) – сохраняет в памяти прямоугольную область экрана в переменной BitMap, которой потом уже нельзя манипулировать как другими переменными – она используется в предикате putimage для вывода сохраненного изображения на экран, при этом координаты X,Y задают местоположение верхнего левого края области. Количество байтов требуемое для сохранения изображения может быть получено с помощью предиката imagesize.

- imagesize(Левый, Верхний, Правый, Нижний, Size)
(integer,integer,integer,integer,string): прототип (i,i,i,i,o) возвращает в Size количесво байтов, необходимых getimage для сохранения изображения. Сохраняемое изображение – это прямоугольник, задаваемый координатами "Левого Верхнего" и "Правого Нижнего" углов. Если для сохранения изображения требуется память больше 64К, то возвращается $FFFF.

- putimage(X,Y,Bitmap, BitOp) (integer,integer,string,integer): прототип (i,i,i,i) помещает изображение, сохраненное с помощью getimage, обратно на экран, при этом (X,Y) являются координатами верхнего левого угла изображения. Аргумент BitOp задает каким образом по цвету существующего пиксела и цвету пиксела выводимого изображения вычисляется цвет каждой результирующей точки на экране:

0 - копируются цвет изображения

1 - выполняется логическое OR (исключающее ИЛИ)

2 - выполняется логическое OR (ИЛИ)

3 - выполняется логическое AND (И)

4 - копируется инверсное изображение.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 1

Последовательность действий

1. Изучить пояснения к работе.

2. Войти в режим редактирования (Alt-E) и ввести определения предикатов для создания на экране трех непересекающихся окон (см. предикат makewindow).

3. Пользуясь средствами Турбо Пролога добавьте предикаты для

- перехода из одного окна в другое;
- очистки окна;
- редактирования текста в текущем окне;
- скроллинга текста в окне;
- удаления окна;
- изменения размеров окна;
- изменения цвета окна и рамки;

4. После ввода каждого нового предиката, программу следует откомпилировать для чего нажать клавиши ALT-F9. Если при компиляции будут обнаружены синтаксические ошибки, то их следует тут же исправить и добиться безошибочной компиляции (см. п. 2.2).

5. Если компиляция прошла успешно, перейти в режим исполнения: Run в главном меню или нажатие клавиш Alt-R.

6. Каждый новый введенный Вами предикат проверьте задавая в окне диалога после подсказки Goal: внешнюю цель с этим предикатом.

7. Включить режим трассировки и просмотреть выполнение предиката по шагам (см. п. 2.3).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 2

1. Инициализировать работу с графикой (предикат initgraph).

2. Построить простые графические объекты: круги, дуги,

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Работа с учебником.

Для обеспечения максимально возможного усвоения материала и с учётом индивидуальных особенностей студента, можно предложить следующие приёмы обработки информации учебника:

- конспектирование;
- составление плана учебного текста;
- тезирование;
- аннотирование;
- составление тематического тезауруса;
- выделение проблемы и нахождение путей её решения;
- самостоятельная постановка проблемы и нахождение в тексте путей её решения;
- определение алгоритма практических действий (план, схема).

В качестве форм и методов контроля внеаудиторной самостоятельной работы студентов могут быть использованы обмен информационными файлами, семинарские занятия, зачеты, тестирование, самоотчеты, контрольные работы, защита творческих работ и электронных презентаций и др.

Контроль результатов внеаудиторной самостоятельной работы студентов может осуществляться в пределах времени, отведенного на обязательные учебные занятия по дисциплине или в специально отведенное время (зачет, экзамен).