В.А. Каймин

Информатика

Учебник

Рекомендовано Министерством образования

Российской Федерации в качестве учебника

для студентов высших учебных заведений,

обучающихся по естественно-научным

направлениям и специальностям

УДК 681.3.06(075.3)

ББК22.18я73

**К 15**

Рецензенты: д-р физ.-мат. наук, профессор, академик *Ю.А. Дубинский,* д-р физ.-мат. наук, доцент *В. Г. Сушко*

Автор: *Каймин Виталий Адольфович,* доктор вычислительных наук, профессор, действительный член Международной Академии Информатизации, вице-президент Всемирного Распределенного Университета

**Каймин В.А.** Информатика: Учебник. - М.: ИНФРА-М,

2000. - 232 с. - (Серия «Высшее образование»).

ISBN 5-16-000170-0

Учебнику присуждено первое место по разделу «Информатика» на конкурсе вузовских учебников Министерства образования Рос­сийской Федерации в 1999 году.

В пособии излагаются теоретические основы информатики, а также техника работы на персональных компьютерах и передачи информации в сети Интернет. Рассмотрены принципы работы сис­тем искусственного интелекта и методы обработки данных на ЭВМ. Приведено большое число задач с примерами решения.

Может использоваться для подготовки к экзаменам по инфор­матике в вузах и школах, а также в системах и учереждениях дис­танционного и дополнительного образования.

Уважаемый читатель!

Перед Вами один из учебников нового поколения по дисциплине «Информатика» для студентов высших учебных заведений, обуча­ющихся по естественнонаучным направлениям и специальностям профессионального образования, написанный известными специ­алистами в области вычислительной техники и прошедший сложный и длительный путь конкурсного отбора.

Данный учебник является одним из двух победителей по дисцип­лине «Информатика» Всероссийского конкурса учебников нового поколения по общим фундаментальным естественнонаучным дисциплинам. Этот конкурс впервые в истории высшей школы в России в связи с реформированием структуры и содержания про­грамм высшего образования был инициирован Госкомвузом России (в дальнейшем - Минобразованием России) и проведен в течение 1995-1998 годов на базе Российского университета дружбы народов.

В конкурсе приняли участие свыше 350 авторских коллективов практически из всех регионов России, заявки представлялись по 11 номинациям, а в их оценке участвовало более ста высококвали­фицированных экспертов.

В результате двух туров конкурса было отобрано 30 авторских коллективов, чьи заявки, а затем и рукописи более всего соответст­вовали как новым учебным программам, так и государственным образовательным стандартам по каждой дисциплине.

Конкурсная комиссия выражает надежду, что данный учебник внесет свой полезный вклад в дело дальнейшего совершенствования российского высшего профессионального образования, и желает всем читателям-студентам и преподавателям - больших творческих успе­хов.

*Заместитель министра образования России,*

*академик Российской академии образования,*

*председатель конкурсной комиссии профессор*

***В.Д. Шадриков***

# Введение

Настоящий учебник посвящен изучению **информатики** - новой научной дисциплины и новой информационной индустрии, связанных с использованием персональных компьютеров и сети Интернет. Раз­витие бизнеса, образования, промышленности и общества в целом учеными, политиками, бизнесменами во многом связывается с широким использованием информационных ресурсов Интернет и на­растающими интеллектуальными возможностями вычислительных машин.

Учебник предназначается студентам вузов и представляет опреде­ленное отражение современного состояния информатики **как науки и информационной индустрии,** в том числе отражение современного состояния персональных компьютеров и возможностей сети Интер­нет как базы для развития современной информационной индуст­рии.

Наиболее распространенным видом **современной вычислительной техники** стали персональные компьютеры IBM PC. По этим причи­нам в учебнике изучаются основные возможности наиболее совре­менных программных средств персональных компьютеров IBM PC - операционной системы Windows, редактора текстов Word, электрон­ных таблиц Excel, базы данных Access, а также телекоммуникационной системы Internet Explourer.

Представление информатики как научной дисциплины связано с рассмотрением проблем **организации вычислений и обработки инфор­мации** с помощью **ЭВМ** и внутри ЭВМ, а также принципов органи­зации и работы машинных интеллектуальных систем (систем искус­ственного интеллекта) на ЭВМ. Для изучения принципов работы систем искусственного интеллекта в учебнике используется язык Пролог - язык моделирования интеллектуальных процессов на ЭВМ.

Изучение **принципов обработки информации и организации вычис­лений** проводится на примерах решения экономических задач, при этом решение задач доводится до получения результатов на ЭВМ путем составления калькуляций, баз данных и программ с парал­лельным анализом и доказательством правильности получаемых ре­зультатов. С этой целью в учебнике излагаются систематические методы безошибочного (доказательного) программирования - но­вого научного подхода к составлению программ, гарантирующих получение правильных результатов с помощью ЭВМ.

Особенностью информатики как учебной дисциплины является **практикум на ЭВМ,** который может проводится в вузе, в школе или дома. Для прохождения такого практикума необходимо иметь пер­сональный компьютер или доступ к нему, а также необходимые пакеты программ - редакторы текстов, базы данных, электронные таблицы и новейшие средства компьютерного обучения - электрон­ные учебники.

Настоящий учебник построен так, что по нему можно изучать информатику даже дома с **использованием персональной ЭВМ,** и по­следующей сдачей зачетов и экзаменов с помощью Интернет. Такая возможность требует наличия на домашних ЭВМ перечислен­ных пакетов программ с операционными системами, редакторами текстов, системами программирования и электронными учебниками.

Особенностью **электронных учебников по информатике,** описанных в настоящем пособии, является то, что с их помощью можно успешно подготовиться к сдаче всех зачетов и экзаменов и сдать их на оценки «отлично» или «хорошо». Основная особенность курса информатики состоит в том, что все экзамены и зачеты по этому курсу, а также контроль знаний должен проводиться на персональных ЭВМ и может производиться дистанционно с помощью сети ЭВМ.

Электронные учебники по информатике были созданы специально для **дистанционного обучения на ЭВМ** с использованием сети Интернет и апробировались в вузах и школах России и Казахстана в течении ряда лет. Сегодня эти электронные учебники можно увидеть на ряде серверов сети Итернет, указанных в настоящем пособии, с помощью любого персонального компьютера, имеющего доступ к Интернет.

Использование этих электронных учебников в ходе апробации систематически приводило к одному и тому же результату - **прак­тически все студенты и школьники** после работы с электронными учебниками успешно сдали все зачеты и экзамены на «хорошо» и «отлично» и создавали при этом стихи, рисунки, базы данных, элект­ронные калькуляции и программы решения на ЭВМ различных задач с доказательствами правильности получаемых результатов.

Особое место в информатике как новой **информационной индустрии** играет Интернет, электронная почта и средства телекоммуникаций, изучение которых вошло в настоящий учебник по информатике. При этом освоение средств Интернет в настоящем учебнике проводится сразу после освоения техники работы и редактирования текстов на персональных ЭВМ.

Опережающее **освоение техники работы в Интернет** позволит сту­дентам и старшеклассникам без задержки войти в сеть Интернет и обратиться в один из центров дистанционного обучения для получения консультаций и помощи в решении текущих и экзаменационных задач у лучших преподавателей.

Благодаря сети Интернат сотни тысяч молодых людей и девушек смогут учиться в лучших **отечественных и зарубежных вузах,** используя персональные компьютеры. Такого рода дистанционное обучение может проводиться без выезда из своего города, поселка или даже не выходя из своего дома.

Первые эксперименты дистанционного и очного обучения с ис­пользованием описанных здесь технологий показали, что **качество обучения** с применением электронных учебников **не хуже,** чем очные занятия в лучших зарубежных вузах, и на порядок лучше, чем при обычной модели обучения без использования каких бы то ни было учебников.

Однако использование Интернет, открывая новые возможности в **предоставлении информации и информационных услуг,** создает и но­вые проблемы. Основные из них - проблемы защиты информации в ЭВМ от искажения, уничтожения и неправомерного доступа.

Ярким примером служит проблема компьютерных вирусов, ко­торые могут уничтожить всю информацию, хранящуюся в ЭВМ. Неправильные действия в сети Интернет, а также копирование программ и данных в сети Интернет может привести к самым не­приятным последствиям. Здесь необходимо твердо усвоить и при­держиваться ряда правил гигиены от компьютерных вирусов.

Согласно **принятому законодательству** информация признана объ­ектом гражданских прав с определением норм и правил отнесения ее к массовой, персональной или конфиденциальной. А базы данных и программы для ЭВМ отнесены к объектам авторского права и на их распространение и использование распространяются соответст­вующие нормы правовой защиты.

Автор выражает свою признательность своим **студентам**, **аспиран­там и сотрудникам,** принимавшим участие в разработках электрон­ных учебников и учебных программ для персональных ЭВМ, а также принимавшим участие в оформлении, редактировании и макетиро­вании учебников по информатике в течении 1985-1999 гг.

*Академик* **В.А.Каимин**

# Глава 1. ИНФОРМАЦИЯ И ПЕРСОНАЛЬНЫЕ ЭВМ

## 1.1. Введение в информатику

**Информатика** - это новая научная дисциплина и новая информа­ционная индустрия, связанные с использованием персональ­ных компьютеров и сетей ЭВМ. В новом тысячелетии предполагается, что основная информация, связанная с деятельностью людей будет храниться в памяти электронных вычислительных машин.

Информатика как **научная дисциплина** изучает законы, принципы и методы накопления, обработки и передачи информации с помощью ЭВМ. В этом смысле информатика как наука является фундаментом для развития новой информационной индустрии, основанной на использовании сетей ЭВМ.

**Фундамент информатики** образуют вычислительные науки - науки об вычислительных процессах и организации вычислительных машин, вычислительных систем и сетей. Основным объектом вычислитель­ных наук являются вычислительные машины - устройства для орга­низации вычислений и обработки символьной информации.

**Обработка, накопление и передача информации** происходит не только внутри ЭВМ. Передачу и накопление информации мы видим при общении людей, в технических устройствах, в живых организмах и в жизни общества, что тоже входит в предмет изучения информатики как научной дисциплины.

**Передача информации** в общении людей - это передача сведений и суждений, данных и сообщений. Даже улыбка является передачей информации при общении людей друг с другом. Любая совместная деятельность людей - работа, учеба и даже игра - построены на обмене и передаче информации.

Для **живых существ** восприятие и передача информации в форме сигналов - основное отличие от неодушевленных предметов окружающего мира. Языковая форма передачи знаковой информации - основное отличие людей от других живых существ.

Слово **информация** происходит от латинского **informatio,** означа­ющего сведения, разъяснения, пояснения. С содержательной точки зрения информация - это сведения о ком-то или о чем-то, а с фор­мальной точки зрения - набор знаков и сигналов.

С юридической точки зрения информация - это сведения о людях, предметах, фактах, событиях и процессах, независимо от формы их представления. Данное определение зафиксировано в **Законе «Об информации, информатизации и защите информации»**, утвержден­ном в 1995 году.

Особую роль для общества играет документированная информация. **Документы** - это информация, зафиксированная на материальном носителе - бумаге или машинном носителе, имеющем реквизиты, позволяющие его идентифицировать.

**Возможность записи** информации в письменном виде - в форме последовательности знаков - привела к образованию государств, возникновению бюрократии и появлению почтовых служб. Парал­лельно это привело - к появлению грамотных людей - людей, уме­ющих читать, писать и искать информацию для решения различных проблем.

**Возникновение письменности** позволило людям не только переда­вать информацию, но и накапливать ее в форме записок, писем и рукописей в архивах, а также в личных и публичных библиотеках. Квалифицированная переработка информации потребовала людей, имеющих надлежащее образование.

**Для обучения грамотности** были открыты гимназии, лицеи и школы, а для подготовки образованных людей - университеты и колледжи, где накоплением и передачей знаний стали заниматься ученые, учи­теля и профессора.

Для **хранения знаний** стали использоваться рукописные книги, а для хранения книг - библиотеки и книгохранилища. В них стал накапливаться интеллектуальный потенциал общества и государств. Неслучайно одной из основных задач варвары считали уничтожение книг и книгохранилищ.

Изобретение **печатных станков** в XV в. создало технологическую основу для массового издания и распространения печатных книг. Это послужило основой для всеобщего распространения грамотности и открытия массовых начальных школ, в которых все дети обучались грамотности - умениям читать, писать и считать.

**Развитие промышленного производства** в XVIII-XIX веках потре­бовало большого числа специалистов, для подготовки которых было открыто большое число университетов. Это дало мощный толчок для развития естественных наук - химии, физики, механики, математики и подготовки инженерных кадров.

Развитие печатных станков привело к **появлению и распростране­нию газет** как средств массовой информации и информатизации общества, а также появлению и распространению журналов для распространения литературных произведений. В это же время появились первые законы, регулирующие авторские права.

Изобретение в XIX - начале XX века **телеграфа, радио и телефона** открыло новые возможности в передаче информации и информати­зации общества. Эти технические средства дали возможность прак­тически мгновенно передавать информацию на любые расстояния.

Следующим шагом технического прогресса стало появление и развитие **электроники, телевидения и радиовещания** к середине XX века. Изобретение телевидения позволило людям видеть на экранах телевизоров события, происходящие в самых различных точках планеты, а изобретение магнитофона - накапливать звуковую и видеоинформацию на магнитных носителях.

Точкой отсчета становления информатики как индустрии стало изобретение в середине XX века **электронных вычислительных машин.** Основной особенностью компьютеров стала возможность автоматической обработки информации. Переработка информации перестала быть исключительной способностью людей и живых существ.

Параллельно в середине XX века были заложены **теоретические основы информатики** как научной дисциплины. В этот период получи­ли развитие математическая логика - фундамент теоретической информатики и теория алгоритмов - фундамент вычислительных наук.

Компьютеры **первого поколения** создавались именно как электрон­ные вычислительные машины для автоматизации сложнейших вычислений оборонного и научного характера. Объем и сложность вычислений, выполнявшихся первыми компьютерами, были недо­ступны даже самым сильным математикам и вычислителям, но посильными для современных домашних компьютеров.

В этот период появились первые **профессиональные программисты** и первые теоретические работы по математической лингвистике, теории искусственного интеллекта и теоретическому программи­рованию. Бурное развитие получили вычислительная и дискретная математика, образующие математическую базу информатики и вы­числительных наук.

Компьютеры **второго поколения** создавались в качестве универ­сальных вычислительных машин, предназначенных для решения задач обработки и накопления информации с использованием устройств ввода и вывода. Компьютеры этого поколения стали использоваться для решения различных научных, экономических, оборонных и инженерных задач.

Для этих машин были созданы **первые операционные системы,** системы программирования и первые диалоговые системы. В этот период программирование зародилось как профессия и появились первые языки программирования и первые инструментальные про­граммы - компиляторы и интерпретаторы для ЭВМ.

**Третье поколение** компьютеров - это первые серийные вычисли­тельные машины для автоматизации обработки и накопления инфор­мации. Для этих ЭВМ был создан целый спектр устройств ввода, вывода и накопления информации. С помощью этих ЭВМ создава­лись первые экспериментальные вычислительные системы и сети.

**Компьютеры третьего поколения** стали широко использоваться в качестве технической базы для самых различных автоматизирован­ных систем - бухгалтерских и банковских систем, банков данных, систем автоматизации проектирования и производства и т. п. В это время появились первые администраторы баз данных и информаци­онные службы по эксплуатации автоматизированных систем.

**Четвертое поколение** - это компьютеры, создаваемые на базе серийных микропроцессоров. С этого поколения ЭВМ началось мас­совое производство и распространение персональных компьютеров, которые могут устанавливаться на любом рабочем столе - дома, на работе или в офисе.

**Персональные ЭВМ** широко используются для учебы, игры, напи­сания писем, книг и отчетов, ведения бухгалтерской документации и экономических расчетов, проведения научных и маркетинговых исследований, сочинения стихов и музыки, ведения переписки с коллегами и друзьями.

**Применение компьютеров** в жизни общества затрагивает условия деятельности и жизни миллионов людей. Современные персональ­ные компьютеры прежде всего открывают возможность выхода в сеть Интернет и оперативного поиска и получения различной инфор­мации в форме электронной почты, электронных журналов, газет и библиотек из самых различных стран и регионов, электронной коммерции - покупок и продаж по всему миру.

В серии ЭВМ четвертого поколения используются и более мощ­ные компьютеры, получившие название **серверов** - вычислительных машин с большим объемом памяти, используемых для постоянного хранения больших объемов информации. Именно такие серверы и используются в качестве узлов связи в вычислительных системах и сети Интернет.

**Академик В.М. Глушков** еще в начале 80-х годов писал, что «к началу следующего столетия в развитых странах основная масса информации будет храниться в памяти ЭВМ, а человек XXI века, который не будет уметь пользоваться ЭВМ, будет подобен человеку XX века, не умевшему ни читать, ни писать».

Обучение компьютерной грамотности - умениям работать с пер­сональными ЭВМ - является основной целью курса информатики в массовой школе. Современное понимание **компьютерной грамот­ности** предполагает не только умения читать, писать и считать с помощью персональных ЭВМ, но и умения искать и передавать информацию с помощью Интернет.

Целью **вузовского курса информатики** считается освоение профес­сионального использования персональных компьютеров и решения на ЭВМ профессиональных задач. Для развития этих умений необ­ходима определенная культура и развитие логического мышления.

Эффективное использование ЭВМ предполагает наличие **инфор­мационной культуры** - умений искать, передавать и подготавливать информацию в форме текстов и рисунков с помощью персональных компьютеров и сети Интернет. Развитие этой культуры ведет к более глубокому развитию логического мышления.

**Логическое мышление** проявляется в умении решать различные интеллектуальные задачи и в том числе в решении сложных задач с помощью ЭВМ. Эти интеллектуальные способности выражаются в умениях рассуждать, доказывать, ставить задачи, а также подбирать и обосновывать способы их решения.

**Сложность изучения информатики** как индустрии связана с ее беспрецендентной динамичностью - технические средства инфор­матики - компьютеры, программы и средства телекоммуникаций полностью модернизируются каждые пять-шесть лет, а соответству­ющие технические знания обновляются каждые два-три года. Одно­временно модернизируются компьютерные сети, архивы, библиотеки и информационные системы.

Подготовка и издание новой учебной литературы требует, как правило, от трех до пяти лет. Поэтому учебная **литература не** **по­спевает за темпами** обновления вычислительной техники. Угнаться за такими темпами обновления техники могут только электронные средства - электронная пресса, электронные справочники, элект­ронные учебники и технологии, связанные с использованием персо­нальных ЭВМ и сети Интернет.

В то же время информатика как научная дисциплина сохраняет свое **ядро - общие принципы, законы и методы,** организации вычис­лений и обработки информации с помощью ЭВМ. Эти принципы сохранят свою роль и значение для всех моделей и типов ЭВМ неза­висимо от их элементной базы, быстродействия и объемов памяти.

Более того **общие законы информатики** как общие законы интел­лектуальной деятельности сохраняют свою силу при изучении прин­ципов обработки, накопления и передачи информации не только в ЭВМ, но и в живых организмах и человеческом обществе.

**В о п р о с ы**

1. Что такое информация?

2. Что такое информатика?

3. Где будет храниться информация в XXI веке?

4. Что такое компьютерная грамотность?

5. Что такое информационная культура?

6. Что такое логическое мышление?

7. В чем сложность изучения информатики?

## 1.2. Персональные компьютеры

**Компьютеры** - это универсальные электронные вычислительные машины (ЭВМ), используемые для накопления, обработки и пере­дачи информации. Самое широкое распространение получили пер­сональные компьютеры, предназначенные для индивидуальной работы.

**Персональные компьютеры -** это малогабаритные вычислитель­ные машины, которые могут быть установлены на любом рабочем месте. Наиболее известны и распространены персональные компью­теры **IBM PC** и **Macintosh.**

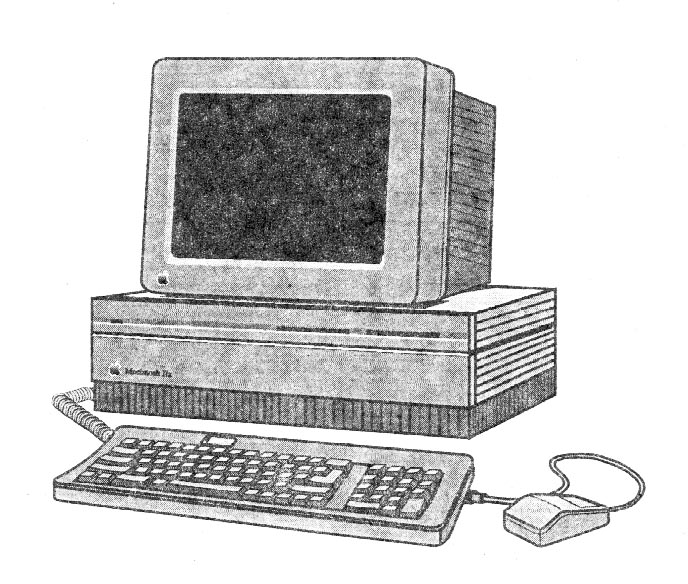
**Минимальный состав** персональных компьютеров:

1) системный блок;

2) дисплей;

3) клавиатура.

**Дисплей** - это устройство отображения информации на элект­ронном экране. Дисплеи в персональных компьютерах могут быть цветными и черно-белыми. Информация на дисплеях обычно отобра­жается как в телевизоре - на экране электронно-лучевой трубки.



*Общий вид персонального компьютера*

**Клавиатура** содержит клавиши, как правило, латинского и рус­ского алфавитов. Кроме того, на клавиатуре имеются цифры и дру­гие специальные знаки. Нажимая на эти клавиши, можно вводить в компьютер самую разную информацию - числа, слова, фразы, а также команды управления компьютером.

**Мышка -** устройство, которое подсоединяется к персональному компьютеру электрическим шнуром и которое можно перемещать по столу.

**Системный блок** содержит процессор и оперативную память. Воз­можности компьютеров зависят от типа и быстродействия процес­сора, а также от объемов оперативной и долговременной памяти. Во всех современных персональных компьютерах в системный блок входят также накопители на магнитных дисках.

**Процессор** - это устройство управления компьютером. Быстро­действие компьютеров определяется числом операций, выполняе­мых процессором за одну секунду. Основной функцией процессоров является автоматическое управление работой ЭВМ с помощью про­грамм, размещаемых в оперативной памяти.

В компьютерах первого поколения быстродействие процессоров составляло несколько тысяч операций в секунду; второго поколения - несколько десятков тысяч, а в машинах третьего поколения - не­сколько сотен тысяч операций в секунду.

**Быстродействие** персональных компьютеров четвертого поколе­ния - несколько миллионов операций в секунду. В компьютерах следующих поколений быстродействие будет составлять десятки и даже сотни миллионов операций в секунду.

В персональных компьютерах IBM PC используются процессоры фирмы Intel. В компьютерах младших моделей процессоры **Intel** - 86, 286, 386 и 486, а в старших моделях процессоры серии **Pentium -** Pentium, Pentium II, Pentium III и т. д. В персональных компьютерах Macintosh применяются процессоры фирмы **Motorola.**

Программа - это последовательность команд и данных, которые могут интерпретироваться ЭВМ. Программы определяют конкрет­ные функции и роли ЭВМ от игрового автомата и редактора текстов до рабочего места президента крупной фирмы или страны.

**Минимальной единицей информации** считается бит. **Бит** - это ве­личина, принимающая значение 0 или 1. Любая другая информация может быть закодирована последовательностью из нулей и единиц. Именно в таком виде вся информация представляется в памяти ЭВМ.

**Единицей памяти** в современных ЭВМ считается байт. **Байты** - это 8-разрядные двоичные числа вида - 00000000, 00000001, ..., 11111111. Один байт записывается в виде 8 двоичных знаков инфор­мации - нулей и единиц:

**1 байт** = 8 бит.

Для измерения памяти большого объема используются следую­щие единицы:

**1 Кбайт** = 1024 байт (1 килобайт);

**1 Мбайт =** 1024 Кбайт (1 мегабайт);

**1 Гбайт** = 1024 Мбайт (1 гигабайт).

Машины первого поколения имели оперативную память порядка нескольких килобайт, компьютеры второго поколения - десятки килобайт, а машины третьего поколения - сотни килобайт.

**Оперативная память** в персональных компьютерах типа IBM PC и Macintosh составляет несколько мегабайт. В больших современных ЭВМ объем оперативной памяти достигает порядка десятков мега­байт, а в компьютерах новых поколений - сотни и тысячи мегабайт.

Для долговременного хранения информации и программ в персо­нальных компьютерах используются **магнитные диски** - гибкие и жесткие. Информация в оперативной памяти удаляется после выключения компьютера. Информация на магнитных дисках может храниться после выключения ЭВМ до следующих сеансов работы.

**Жесткие диски** - это устройства хранения информации, программ и данных в ЭВМ. В персональных компьютерах жесткие диски нахо­дятся внутри системного блока и служат для постоянного хранения программ, данных, архивов и т.п.

**Объем памяти** на жестких дисках в современных компьютерах имеет диапазон от нескольких мегабайт до нескольких гигабайт. В компьютерах новых поколений объем памяти на жестких магнит­ных дисках будет составлять десятки и сотни гигабайт.

**Гибкие диски** - это сменные носители информации, на которых программы и данные можно хранить отдельно от ЭВМ. Гибкие дис­ки используются для личного хранения и переноса программ и дан­ных от одного компьютера к другому. Объем памяти на наиболее широко распространенных гибких магнитных дисках составляет от 360 Кбайт до 1,4 Мбайт.

К современным персональным компьютерам может быть подсо­единен целый **ряд дополнительных устройств.** Наиболее часто к ним подключаются принтеры, модемы и компакт-дисководы. Компакт-дисковод - это устройство для считывания компакт-дисков.

**Компакт-диск** - это оптические диски с голографической записью информации. Особенность компакт-дисков - большой объем запи­санной на них информации, равной объему порядка 500 гибких дисков.

**Компакт-диски** - это средство для постоянного хранения инфор­мации, которая записывается один раз и может многократно считы­ваться на ЭВМ. Компакт-диски - наиболее удобное средство для переноса больших объемов информации. Объем памяти на компакт-дисках составляет до 780 Мбайт.

**Принтер** - это печатающее устройство, подсоединяемое к компь­ютерам. Наибольшее распространение получили три типа принте­ров, различающихся скоростью и качеством печати: матричные, струйные и лазерные. Самые простые и дешевые среди них - мат­ричные, самые быстрые и качественные - лазерные, а струйные - самые качественные среди дешевых принтеров.

**Модем** - это устройство передачи информации по линиям теле­фонной связи. С помощью модемов персональные компьютеры могут подключаться через телефонную сеть к другим компьютерам, а также входить в различные телекоммуникационные компьютер­ные сети.

**Скорость передачи** информации по линиям связи оценивается в бодах и килободах. Скорость в один бод - это передача одного бита в секунду:

**1 бод** = 1 бит/секунда.

**1 Кбод** = 1024 бод.

**В о п р о с ы**

1. Какие устройства входят в состав персональных компьютеров?

2. Что такое процессор?

3. Каково быстродействие современных процессоров?

4. В каких единицах измеряется объем памяти компьютеров?

5. Каков объем оперативной памяти современных компьютеров?

6. Каковы объемы памяти на гибких дисках?

7. Каковы объемы памяти на жестких дисках?

8. Каковы объемы памяти на компакт-дисках?

## 1.3. Работа на персональных ЭВМ

**Работа на ЭВМ** обычно проходит в форме диалога человека с компьютером. Человек просматривает информацию на экране компьютера, указывает на нее мышкой, нажимает клавиши, набирает команды, вводит слова, числа, фразы и т. п. В ответ компьютер выводит свою информацию: сообщения, меню, заставки, диаграммы, рисунки, результаты вычислений и обработки данных.

Работа ЭВМ основана на использовании программ. **Программы для ЭВМ** - это форма представления данных и команд, предна­значенных для получения определенных результатов или способа функционирования ЭВМ.

**Совокупность программ** для данного типа ЭВМ определяет все многообразие их применений. На персональных компьютерах на­иболее часто применяются игры, редакторы текстов, базы данных, информационные системы, электронные таблицы, системы програм­мирования и т. п.

Главной среди программ на ЭВМ является **операционная система,** которая постоянно хранится в долговременной памяти компьютера. Работа ЭВМ начинается с загрузки операционной системы, а все ос­тальные программы запускаются с помощью операционной системы.

**Операционная система** - это главная программа, управляющая работой компьютера в целом. На персональных компьютерах типа IBM PC используются в основном операционные системы **MS DOS** и **Windows.** В персональных компьютерах Macintosh применяется операционная система OS/7.

Операционная система **MS DOS** - это самая простая операцион­ная система для компьютеров IBM PC. Она используется на всех младших моделях IBM PC и может применяться на всех старших моделях компьютеров этого же типа.

Операционная система **Windows** - наиболее современная и удоб­ная операционная система для старших моделей персональных компьютеров IBM PC. Эта система может использоваться только на компьютерах старших моделей с оперативной памятью более 2 Мбайт и памятью на жестких дисках не менее 80 Мбайт.

На персональных компьютерах IBM PC используются несколько версий операционной системы Windows; созданных всемирно извест­ной фирмой Microsoft: Windows 3.1, Windows-95, Windows-98, Windows-2000, отличающихся своими функциями и возможностями.

**Основными объектами** во всех операционных системах на ЭВМ являются файлы, программы и каталоги. Все программы в ЭВМ пред­ставляются отдельными файлами или наборами файлов, хранящихся в определенном каталоге.

**Файлы** - это последовательность записей на машинных носите­лях - магнитных или оптических дисках, магнитных или перфо­лентах и т.п. Все данные и программы на ЭВМ записываются в виде файлов или наборов файлов. Все файлы в памяти ЭВМ имеют уникальные имена.

**Совокупности файлов** в памяти ЭВМ объединяются в форме ката­логов и подкаталогов. Каждый каталог имеет свое уникальное имя. Имя подкаталога образуется из его собственного имени и имени каталога, в котором он находится. Имена каталогов (оглавлений) записываются большими (прописными) буквами, а имена файлов - малыми (строчными) буквами.

В операционных системах MS DOS и Windows **имена файлов** об­разуются из латинских букв и цифр с добавлением трехбуквенных окончаний после точки. Для записи окончаний в этих операцион­ных системах приняты правила:

.ехе - программа, готовая к выполнению;

.com - программа, готовая к выполнению;

.bat - командный файл операционной системы;

.txt - текстовый файл;

.doc - текстовый файл.

**Работа с любыми операционными системами** - это в основном работа над каталогами файлов и программ, размещенных на магнит­ных и оптических дисках. Эта работа состоит в просмотре каталогов и подкаталогов, копировании файлов и запуске тех или иных про­грамм.

В любой современной операционной системе работа с ЭВМ про­исходит в основном с помощью менеджеров программ и файлов. Эта программа позволяет человеку в диалоге с компьютером просматри­вать каталоги программ и файлов во внешней памяти:

С: А:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Name | Name | Name | Name | Name |
| … | begin.bat | file2.dat | TUTOR | MUSIC |  |
| DOS | prog1.exe |  | TEACHER | GRAF |  |
| NC | prog2.exe |  | CALC | DOC |  |
| KAYMIN | text1.txt |  | EDIT |  |  |
| BOOK | text2.txt |  | BASE |  |  |
| PAСET | file1.dat |  | PROLOG |  |  |
| text1.txt | 4096 | 26.12.96 | 6720 bytes in 1 files selected | | |

В приведенном примере указаны два **каталога:** каталог на жест­ком диске С, на котором размещена операционная система DOS, и каталог на гибком диске А с пакетом программ, включающим электронный учебник TEACHER, клавиатурный тренажер TUTOR и другие учебные программы из пакета программ для лабораторных работ по информатике.

**Запуск программ** на персональных ЭВМ обычно проводится пе­ремещением курсора на экране с помощью клавиш-стрелок или мышки на имя программы в каталоге, подлежащей выполнению, а затем - нажатием клавиши ввода Enter на клавиатуре либо нажати­ем клавиши на мышке.

Для **установки новых программ** на ЭВМ они должны быть предва­рительно записаны на оптическом или гибких дисках, либо получе­ны по электронной почте через Интернет. Для этого диск с новыми программами должен быть установлен в соответствующий дисковод.

Для **перезаписи** программ и файлов необходимо выявить каталог, где они записаны; далее указать или создать каталог, куда они должны быть переписаны и только после этого указать команду **«запись»** и нажать клавишу Enter либо клавишу на мышке.

В любом случае до записи новых программ на жесткие диски вашей ЭВМ необходимо **проверить отсутствие вирусов** на этих дисках или в файлах, полученных по Интернет. При обнаружении вирусов немедленно пролечите эти файлы и диски с помощью антивирусных программ.

**Компьютерные вирусы -** это специальные саморазмножающиеся программы. Эти программы могут испортить или уничтожить про­граммы и файлы, хранящиеся в памяти компьютера. Заражение компьютерными вирусами происходит исключительно при копиро­вании файлов с помощью дискет или при их передаче по сети Интернет.

**Наиболее опасные вирусы** могут испортить или уничтожить всю информацию в ЭВМ и сделать неработоспособным компьютер и даже сеть ЭВМ. Для предотвращения от таких последствий используются специальные меры и программы.

Для защиты от вирусов используются специальные **антивирусные программы,** которые необходимо устанавливать и периодически обновлять на ЭВМ. Работа антивирусных программ заключается в диагностике и удалении компьютерных вирусов в файлах и програм­мах на ЭВМ.

Для надежной работы Ваших компьютеров и предотвращения потерь информации на дисках придерживайтесь следующих правил **«компьютерной гигиены»:**

1) при вводе чужих дисков в свою машину первым делом про­верьте их на вирусы;

2) после работы на чужой машине сразу же проверьте свои диски на вирусы.

Для защиты от компьютерных вирусов при работе в сети Интернет придерживайтесь следующих правил:

3) не открывайте файлы, полученные по почте, без проверки их на вирусы;

4) не запускайте программы, полученные по Интернет, без их про­верки на вирусы.

Для борьбы с компьютерной заразой постоянно обновляйте свои антивирусные программы. Учтите, что «супер-программисты», со­здающие компьютерные вирусы, постоянно ищут средства и способы для преодоления защиты от вирусов и проникновения в чужие компьютеры.

Кроме защиты от вирусов на любых ЭВМ может быть предусмот­рена также защита от несанкционированного доступа к ЭВМ и хра­нящимся в них данным. Простейшим средством для этого является введение паролей для доступа к ЭВМ или определенным сегментам ее памяти.

**В о п р о с ы**

1. Что такое - программы для ЭВМ?

2. Каковы основные типы программ?

3. Что такое операционная система?

4. Какие операционные системы используются на IBM PC?

5. Что такое файл на ЭВМ?

5. Как записываются имена файлов и каталогов на ЭВМ?

6. Как проводится перезапись программ на компьютере?

7. Что такое компьютерные вирусы?

8. Как предохраняться от компьютерных вирусов?

9. Для чего используются пароли на ЭВМ?

**З а д а н и е**

1. Проработайте на компьютере с клавиатурным тренажером следу­ющие наборы слов:

а) русские слова; в) английские слова;

б) числа и формулы; г) слова языка Бейсик.

2. Узнайте, как на Вашем компьютере запускаются антивирусные программы.

3. Проверьте память своего компьютера на наличие вирусов.

4. Проверьте свои диски на наличие компьютерных вирусов.

5. Узнайте - сколько свободной памяти на Ваших дисках.

## 1.4. Редактирование текстов на ЭВМ

**Подготовка и редактирование текстов** - это одно из наиболее частых применений персональных компьютеров. Простота и удобство редактирования текстов на ЭВМ привели к тому, что для подобных работ практически перестали использоваться пишущие машинки.

**Редакторы текстов** на ЭВМ - это специальные программы, по­зволяющие вводить, искать, редактировать и сохранять различные тексты на ЭВМ вплоть до научных отчетов и литературных произве­дений, а также личных и служебных архивов.

Редакторы текстов на ЭВМ - это **наиболее удобное средство** для подготовки различного рода документов и создания архивов доку­ментов на ЭВМ. Ведение такого рода архивов составляет основу работы большого числа людей - бизнесменов, юристов, писателей, ученых, журналистов, инженеров, секретарей, референтов и многих других специалистов.

**Документом** считается информация, зафиксированная на матери­альном носителе, имеющем реквизиты, позволяющие его иденти­фицировать. К числу реквизитов документа относится фамилия автора (исполнителя) и дата его создания (подписания), а также входящая или исходящая регистрация при размещении его в архивах.

На персональных компьютерах IBM PC наибольшее распростра­нение получили редакторы текстов Word и Лексикон. **Word** - это лучший редактор текстов для операционной системы Windows. **Лексикон** - это один из лучших отечественных редакторов текстов для компьютеров с операционной системой MS DOS.

В ЭВМ с накопителями на жестких магнитных дисках могут хра­ниться и **редактироваться целые книги.** Одна страница текста, име­ющая 30 строк по 60 знаков в строке, требует для хранения 1800 байт ≈ 1,76 Кбайт памяти.

**Книга из 100 страниц** указанного размера занимает около 176 Кбайт на магнитных дисках. Соответственно, на дисках объемом 200 Мбайт может храниться более 100 таких книг.

Набор текстов на клавиатуре компьютера проводится так же, как и на пишущих машинках. Но при этом тексты выводятся не на бумагу, а на экран дисплея, на котором и производятся все исправ­ления. Если текст большой, то на экране будет видна только его часть, а весь текст будет храниться в памяти ЭВМ.

Экран ЭВМ:

Я помню чудное мгновенье:

Передо мной явилась ты,

Как мимолетное виденье,

Как гений чистой ⇐

Значок **⇐** для указания на экране места исправления символа или слова называется **курсором.** Перемещение курсора по экрану про­водится с помощью мышки или клавиш стрелок. Для исправления букв, слов или фраз курсор подводят к их началу.

**Исправления в тексте** на персональных компьютерах можно вно­сить неоднократно. Вставка или замена символов и слов проводится набором их на клавиатуре. Удаление символов и слов выполняется нажатием клавиши Del или Bs. Для переключения режима вставка/замена нажимается клавиша Ins. Удаление строк и вставка новых строк проводятся выделением их на экране с последующим нажатием клавиши Del.

Используя **выделение фрагментов** текстов на экране компьютера, их можно переносить из одного места текста в другое. Можно выде­лять их курсивом, подчеркиванием, изменением толщины или вида шрифта.

**В многооконных редакторах** фрагменты можно переносить из одного текста в другой. Для этого на экране одновременно открыва­ется два или более текстов сразу в нескольких окнах.

**Поиск текстов** на магнитных дисках также проводится через ос­новное меню указанием на слово «файл» (file), а затем на режим «открытие» (open). Результатом будет появление оглавления с име­нами файлов и других каталогов, на которые можно указывать с помощью мышки или клавиш-стрелок. Нажатие клавиши Enter приведет к появлению текста на экране ЭВМ.

**Запись текстов** на диски в редакторах проводят обращением к основному меню, указав на слово «файл» (file). Далее в появившемся на экране подменю нужно указать вид операции - «запись»(write), а затем имя файла, под которым текст записывается на магнитные диски, после чего нажимается клавиша ввода Enter.

Во многих редакторах можно создавать **сложные тексты,** вставляя различные таблицы, диаграммы, рисунки и даже фотографии. С помощью этих средств можно создавать письма, документы и от­четы, сохраняя их на магнитных дисках и печатая в необходимом количестве экземпляров.

Возможности современных текстовых редакторов на персональ­ных ЭВМ таковы, что с их помощью можно выполнять не только редакционную подготовку документов, писем и отчетов, но и **изда­тельскую подготовку** газет, журналов и книг практически в домашних условиях.

Создание книг на ЭВМ приводит к появлению **электронных книг** и **журналов** как новых видов литературных произведений. Копи­рование и распространение электронных книг, газет и журналов должно подчиняться законам авторского права, так же как и созда­ние и распространение бумажных книг, журналов и газет.

**Авторские права** на произведения состоят в следующем:

1) право на имя - в произведении должно присутствовать имя автора;

2) право на изменения - только автор может вносить изменения в произведение;

3) имущественные права - автор является собствеником создан­ного произведения.

**Автором** считается лицо, творческим трудом которого создано произведение. Автор имеет права требовать указания своей фамилии на всех экземплярах (копиях) своего произведения. В соответствии с законом об авторских правах только автор может вносить измене­ния при модификации произведения.

**Защита прав** собственности на произведение фиксируется знаком © (copyright - права на копирование) с указанием фамилии или псев­донима автора. Включение в произведение копирайта (знака ©) означает, что никто не имеет права копировать произведения без заключения письменного договора с его автором (авторами).

Согласно **международному праву** на электронные книги, учебни­ки, базы данных и программы для ЭВМ распространяются те же авторские права, как и на обычные литературные произведения.

Использование программ, баз данных, электронных книг и учеб­ников в коммерческих целях возможно только при заключении договоров с владельцами авторских прав. **Нарушители авторских прав** обязаны возмещать ущерб издателям и авторам произведений.

**В о п р о с ы**

1. Что такое редактор текстов?

2. Что такое - документ?

3. Каковы основные операции редактирования текстов?

4. Сколько страниц текста может быть записано:

а) на кассете магнитной ленты с объемом памяти 100 Кбайт;

б) на гибком диске с объемом памяти 1 Мбайт?

5. Как тексты записываются на магнитные диски?

6. Как узнать, какие тексты хранятся в памяти компьютера?

7. Что такое авторские права на произведение?

8. В чем состоят имущественные права авторов?

9. Как называются нарушители авторских прав?

**З а д а н и я**

1. Введите с помощью редактора текстов любимое четверостишие, выведите его на бумагу и запишите на магнитные диски.

2. Подготовьте текст поздравления с Новым годом для родителей, близких и друзей. Выведите на печать два-три поздравления для разных адресатов.

3. Перепишите четверостишие и поздравления на ваши диск и по­пробуйте прочитать их с помощью диспетчера файлов или программы **Norton Commander.**

4. Подготовьте на ЭВМ договор на издание сборника ваших расска­зов с указанием Ваших авторских прав.

## 1.5. Контроль знаний на ЭВМ

**Контроль и проверка** знаний при дистанционном, заочном и очном обучении может производиться с помощью персональных ЭВМ. В этих целях создаются и используются специальные компьютерные программы, получившие название электронных учебников.

**Электронные учебники -** это компьютерные программы, содер­жащие учебные тексты и тесты. Тесты - это контрольные вопросы. Тесты в электронных учебниках отличаются тем, что ответы на них проверяются ЭВМ непосредственно в диалоге с учащимися.

**Основные возможности** электронных учебников состоят в следу­ющем. Электронные учебники могут использоваться:

1) для проверки знаний на занятиях с использованием ЭВМ;

2) в качестве экзаменатора на зачетах и экзаменах;

3) в качестве тренажеров на самостоятельных занятиях с ЭВМ.

Примером может служить **электронный учебник по информатике,** являющийся приложением к настоящему бумажному учебнику. Этот электронный учебник используется для контроля знаний студентов и школьников по информатике на зачетах и текущих занятиях, а также при подготовке и сдаче экзаменов.

Особенностью данного электронного учебника является его **связь с бумажным учебником.** Все тексты электронного учебника взяты из настоящего учебника и все ответы на тесты электронного учебника можно найти в данном учебном пособии.

Электронным учебником по информатике можно пользоваться **на компьютерах IBM PC** с операционными системами MS DOS и Windows. В среде MS DOS запуск электронного учебника осуществ­ляется из каталога TEACHER, в котором находится программа, с помощью команды

**> menu**

Если запуск прошел успешно, то на экране появится заставка с названием программы, а затем - окно регистрации, в котором не­обходимо указать свою фамилию, имя и группу:

**Информация Регистрация Работа Темы Протокол Выход**

**Зарегистрируйтесь:**

**Фамилия: \_\_\_\_\_\_\_\_\_\_\_**

**Имя: \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Группа: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата: 01/01/1998**

После регистрации на экране появляется основное меню элек­тронного учебника. Работа с электронным учебником проходит в режиме диалога с ЭВМ в трех основных режимах:

**1) чтение** разделов учебника;

2) **обучение** с использованием тестов;

**3) контроль** знаний учащихся.

В режиме **«чтение»** на экране можно читать разделы электронного учебника, организованные в главы и разделы точно так же, как и в бумажном учебнике. Пример **оглавления:**

**Информация Регистрация Работа Темы Протокол Выход**

**Иванов**

**Глава 1. Информация и персональные ЭВМ**

**Глава 2. Элементы информационных технологий**

**Глава 3. Элементы «искусственного интеллекта»**

**Глава 4. Алгоритмы и начала программирования**

**Глава 5. Технология решения задач на ЭВМ**

Пример текста из первого раздела учебника:

**Информация Регистрация Работа Темы Протокол Выход**

**Иванов**

**Глава 1.1. Информация и персональные ЭВМ**

**1.1.1. Введение в информатику**

**Информатика** - это научная дисциплина, изучающая законы и методы накопления, обработки и передачи ин­формации с помощью ЭВМ. **Накопление, передача** и **обработка информации** происходит не только внутри ЭВМ, но и при общении людей, в технических устрой­ствах, в живых организмах и в жизни общества.

В **режиме чтения** тексты можно перелистывать с помощью кла­виш стрелок и клавиш пролистывания страниц вверх-вниз. Смена темы проводится через основное меню, расположенное на верхней строке экрана.

**Контрольные вопросы** можно получить в режимах контроля и обу­чения. Разница между ними заключается в том, что в режиме кон­троля на экране появляется только сам контрольный вопрос, а в режиме обучения на экране можно читать и перелистывать текст соответствующего раздела.

Пример контрольного вопроса:

**Информация Регистрация Работа Темы Протокол Выход**

**Иванов**

**Глава 1.1. Информация и персональные ЭВМ**

**1.1.1. Введение в информатику**

**Контрольный вопрос**

***Информация -* [\_\_\_\_\_] о ком-то**

**или о [\_\_\_\_\_], передаваемые**

**в форме [\_\_\_\_] и сигналов.**

**Ответы на контрольные вопросы** вводятся с клавиатуры в места соответствующих пропусков. Слова могут вводиться в любом порядке и при необходимости тут же исправляться. Оценка ответа будет выведена ЭВМ на экран сразу после нажатия клавиш Ctrl и Enter.

Ответ считается правильным, если получится **фраза, выражающая истинное утверждение.** Эталонные правильные ответы можно найти в соответствующем разделе бумажного или электронного учебника.

ЭВМ выводит на экран оценку **«молодец»,** если ответ полностью совпадет с соответствующей фразой в учебнике. Оценка **«хорошо», «нехорошо»** и **«плохо»** будет выводиться ЭВМ в зависимости от числа совпадений и количества введенных слов.

Оценка **«хорошо»** ставится, если не менее 50% слов совпадают с эталоном из учебника. В случаях, когда более половины слов отли­чается от эталонных ответов, выводится оценка **«нехорошо».** Оценка «плохо» выводится, если не введено ни одного слова.

В режиме обучения **правильный ответ** к контрольному вопросу можно найти с помощью подсказки, нажав клавишу F6. В результате на экране появится раздел учебника с эталонным ответом, который можно найти перелистывая текст раздела. Возврат к контрольному вопросу произойдет после повторного нажатия клавиши F6.

**Все контрольные вопросы** составлены исключительно по матери­алам учебника и в нем всегда можно найти все ответы на все кон­трольные вопросы. Поэтому в режиме обучения, используя доступ к текстам учебника, при внимательной работе можно получать исклю­чительно отличные и положительные оценки.

Режим обучения особенно удобен при **самостоятельной подготовке** к зачетам и экзаменам. Именно при работе в этом режиме могут быть проработаны все контрольные вопросы по заданному разделу учебника и по всему учебнику в целом.

**Педагогический эффект** от работы с рассматриваемыми тестами заключается в том, что заполнение тестов требует нахождения, запо­минания, проговаривания и ввода слов-ответов с клавиатуры. При этом при анализе ответа необходимо осмысление подбираемых фраз на предмет их истинности.

Работа с электронными учебниками позволяет не только мгно­венно сообщать оценки каждого теста учащемуся, но и работать в удобном темпе, используя тексты учебника и получая подсказки от ЭВМ. При этом с компьютером можно заниматься столько - сколько нужно для **получения положительных или отличных оценок.**

Работу по контролю знаний на традиционных **занятиях в школах и вузах** стараются выполнять учителя и преподаватели. Однако при групповом обучении они не могут опросить в течении занятия всех учащихся и поэтому они не могут дать гарантий в результатах обуче­ния всем своим ученикам.

Определенным выходом для традиционных форм обучения могло бы служить **использование бумажных тестов,** которые позволяют проверять знания всех учащихся по любому из текущих занятий. Однако проверка этих бумажных материалов весьма трудоемка для преподавателей и они минимизируют или игнорируют эту работу.

**Компьютерный контроль знаний** позволяет освободить преподава­телей и учителей от весьма трудоемкой и рутинной работы по про­верке всех результатов тестирования. Эту работу на себя вполне могут взять персональные компьютеры, снабженные соответствующими электронными учебниками.

Однако **наиболее эффективно обучение** с использованием электрон­ных учебников происходит при контроле и руководстве преподава­телей, которые организуют систематический контроль знаний и ре­шение задач по всему курсу.

**Проверка** знаний в представляемом электронном учебнике про­исходит в режиме «контроль». В этом режиме на экран выводятся только тесты по выбранной теме без каких либо подсказок. Оценка ответа выводится на экран ЭВМ сразу же после его ввода с клави­атуры.

При этом все ответы **записываются в протокол** на магнитных дис­ках, который можно увидеть на экране по завершении серии тестов по выбранной теме. По завершении серии тестов на экран выводятся итоги прохождения проработанной темы в следующей форме:

-----------------------------------<ИТОГИ>------------------------------------

***>* Количество тестов =15**

**> Отлично =11**

**> Хорошо = 3**

**> Нехорошо = 1**

**> Плохо = 0**

**>!> Совпало ответов = 87 % <!<**

**>!> 0тсутствуют ответы = 0 % <!<**

**-----------------------------------------------------------------------------------**

**Окончательных оценок** компьютер не ставит. Задача компьютера состоит в сборе статистики ответов и ведении протокола ответов в режиме «контроль». Оценка ответов выводится учащимся на экран сразу после проработки серии тестов по текущей теме.

**Проставление окончательных оценок** остается за преподавателями, которые должны просматривать протоколы ответов с учетом возмож­ных альтернативных правильных ответов учащихся. Если ответ уча­щегося правильный, но отличается от эталонного ответа в учебнике, то преподаватель обязан скорректировать окончательную оценку за этот ответ и итоговые оценки за работу в целом.

**Протокол ответов** можно увидеть на экране через главное меню, не завершая работы с электронным учебником. Пример протокола диалога с компьютером при работе с электронным учебником по информатике:

**Фамилия: Чуков**

**Имя: Андрей**

**Группа: 434**

**Дата: 10/01/1997**

###################################

**Режим: Контроль**

###################################

***>@* Глава 2. Элементы информационных технологий.**

**>@ 2.4. Базы знаний на ЭВМ.**

**Признак - логическая [характеристика]**

**[объекта] / субъекта / [процесса].**

**------------------------------------------------------------------------------------**

**Определение понятия - совокупность [признаков],**

**характеризующих [содержание] понятия.**

**------------------------------------------------------------------------------------**

**Содержание понятия - совокупность [признаков],**

**выделяющих {объект}, отвечающих данному**

**понятию, [среди] других [объектов].**

------------------------------------------------------------------------------------

--------------------------------------<**ИТОГИ**>--------------------------------

**> Количество тестов =15**

**> Отлично =11**

**> Хорошо = 3**

**> Нехорошо = 1**

**> Плохо = 0**

**>!> Совпало ответов = 87 % <!<**

**>!>Отсутствуют ответы = 0 % <!<**

**----------------------------------------------------------------------------**

**<СИСТЕМА ЗАКОНЧИЛА РАБОТУ>\*<10/01/1995>**

Такого рода протоколы в виде файлов формируются и автомати­чески записываются на магнитные диски. Эти протоколы накапли­ваются и хранятся на ЭВМ. При дистанционных формах обучения эти протоколы в виде файлов могут пересылаться по электронной почте и служить подтверждением результатов проработки учебных материалов.

**В о п р о с ы**

1. Что такое - электронные учебники?

2. Что такое - выборочные ответы?

3. Когда ответ считается правильным?

4. Когда и как получить подсказки?

5. Как добиться хорошей успеваемости?

**З а д а н и я**

1. Зарегистрируйтесь в электронном учебнике.

2. Сравните оглавления электронного и бумажного учебников.

3. Пролистайте первый раздел электронного учебника.

4. Проработайте тесты первого раздела учебника.

5. Ответьте на контрольные вопросы первого раздела.

# Глава 2. ЭЛЕМЕНТЫ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

## 2.1. Основные возможности Интернет

Новым явлением в жизни общества в конце XX века стало созда­ние глобальной **международной компьютерной сети** Интернет. Эта компьютерная сеть для многих людей стала средством оперативного получения самой различной информации - электронных копий газет, писем, журналов, книг, программ, игр и информационных новостей.

**Передача и получение информации** с помощью сети Интернет за рубежом стали самым массовым применением персональных компь­ютеров. Оперативное получение разнообразной информации дома и на работе с помощью персональных ЭВМ становится доступным после подсоединения их к телефонной или телекоммуникационной сети.

**Вычислительная сеть** - это объединение ЭВМ, в которой каждый компьютер может передать информацию любому другому компью­теру, подключенному к сети. Главной возможностью вычислительных сетей является доступ к информации, документам и информацион­ным ресурсам, хранящихся на компьютерах, подключенных к сети.

**Информационные ресурсы** - совокупность документов в архивах, библиотеках, фондах, банках данных и других информационных системах. В вычислительных сетях информационные ресурсы, хра­нящиеся в ЭВМ, могут быть доступны с помощью других персо­нальных компьютеров, подключенных к этой сети.

По количеству объединяемых компьютеров сети подразделяются на **локальные, корпоративные, региональные** и **глобальные.** Соответст­венно можно говорить о корпоративных, региональных и глобальных информационных ресурсах, доступных в этих сетях ЭВМ.

**Локальная вычислительная сеть** - это сеть, связывающая компью­теры в одном или нескольких соседних помещениях. Примерами локальных сетей являются кабинеты учебных компьютеров в школах, вузах, офисах и лабораториях.

Ведущую роль в организации вычислительных систем и сетей играют серверы. **Серверы** - это мощные компьютеры с большой дисковой памятью, используемые для хранения файлов и программ. Серверы применяются для накопления и передачи общедоступной информации в вычислительных сетях и работают обычно 24 часа в сутки.

При этом на уровне локальных сетей возникают вопросы органи­зации **надежного хранения информации** и ее защиты от несанкциони­рованного доступа. Для ведения, надежного хранения и защиты информации в сети привлекаются специалисты, профессия которых получила название - сетевые администраторы.

**Региональная вычислительная сеть** объединяет компьютеры в пределах определенного региона. Примерами региональных сетей являются корпоративные сети ЭВМ в банках и корпорациях, а также в промышленных и коммерческих предприятиях.

**Для подключения** к региональной сети персональных компьютеров необходимы специальные средства связи - каналы связи, модемы и сетевое программное обеспечение. В качестве средств телекомму­никаций между компьютерами используются обычные телефонные каналы, а также специальные оптоволоконные линии связи либо спутниковые каналы радиосвязи.

**Модемы** - это устройства связи ЭВМ с другими компьютерами с использованием каналов связи. Функции модемов состоят в преобра­зовании данных, представляемых в ЭВМ в виде 0 и 1, в телефонные сигналы с двумя несущими частотами. Основной характеристикой модема является скорость передачи данных, измеряемой в бодах.

**Скорость**, наиболее распространенных современных модемов со­ставляет 2400, 9600, 14400, 19200, 22800 и 33600 Бод. Для электрон­ной почты можно использовать любой из этих модемов. Для работы в сети Интернет требуются модемы со скоростью передачи не ниже 19200 Бод.

На уровне регионов и крупных корпораций **вопросы защиты** и надежного ведения информации, образующей корпоративные ин­формационные ресурсы, является серьезной проблемой. Ошибки в проектировании и эксплуатации корпоративных информационно-вычислительных сетей могут привести к серьезному экономическому ущербу.

**Глобальная сеть** объединяет компьютеры, расположенные в раз­личных странах на различных континентах земного шара. Пример глобальной сети ЭВМ - международная компьютерная сеть Интер­нет. Для подключения к сети Интернет используются специальные сетевые программы.

**Сетевые программы** - это программы получения доступа к ин­формации, информационным ресурсам и информационным сис­темам, используемым в вычислительной сети. Примерами таких программ является сетевые пакеты **Internet Explourer** фирмы Microsoft и **Netscape Navigator** фирмы Netscape, созданные для рабаты на компьютерах IBM PC с операционной системой Windows.

Применительно к сети Интернет используется понятие **глобаль­ных информационных ресурсов,** доступных из любой точки нашей планеты. Развитие сети Интернет является базой для организации хранения и доступа ко всей информации, накапливаемой челове­ческим обществом.

Основой современной информационной компьютерной индустрии и сети Интернет является всемирная распределенная сеть электрон­ных библиотек **WWW - World Wide Web.** Электронные библиотеки в этой сети размещены на специальных серверах.

**Электронные библиотеки** в сети Интернет организуются с по­мощью гипертекстов. В форме гипертекстов могут храниться самые различные документы, газеты, журналы, каталоги, прейскуранты, а также всевозможные информационные и информационно-поиско­вые системы и сетевые программы - сетевые компьютерные игры, сетевые электронные учебники, сетевые справочники и энциклопе­дии.

**Гипертекст** - это совокупность страниц с текстами, картинками и ссылками на другие страницы. Ссылки могут делаться как на стра­ницы данного гипертекста, так и на страницы любого другого гипер­текста, хранящегося в данном компьютере либо даже на любом другом файл-сервере, зарегистрированном в сети Интернет.

Для доступа к электронным библиотекам используются специ­альные **гипертекстовые имена,** выполняющие роль адреса сервера в сети. Коды, указываемые в адресах, утверждены **международным стандартом.** Пример - гипертекстовое имя в сети Интернет сервера виртуального учебного центра, в котором установлена электронная копия настоящего учебника по информатике вместе с системой контрольных тестов:

**www. prometеy. ankey.ru**

В рассматриваемом примере имя сервера состоит из четырех час­тей. Первое слово **www** - это признак подключения сервера к сети Интернет. Второе имя **prometey** - это имя системы дистанционного обучения Прометей. Третье имя **ankey** - это имя корпорации Анкей, которой принадлежит данный сервер. Последнее четвертое слово **ru** - это идентификатор сектора России в сети Интернет.

Примерами электронных библиотек в сети Интернет могут слу­жить серверы различных центральных газет, журналов - сервер га­зеты **Известия,** сервер журнала **Итоги,** сервер радиостанции **Русское Радио.**

В Интернете Вы можете найти несколько игровых серверов, а также серверы Центров дистанционного обучения ведущих москов­ских и российских вузов - **МЭСИ** (**М**осковский государственный университет **Э**кономики, **С**татистики и **И**нформатики) и **МИЭМ** (**М**осковский **И**нститут **Э**лектроники и **М**атематики).

**Электронная почта** - это способ передачи писем с помощью пер­сональных компьютеров и средств телекоммуникаций. В качестве писем по электронной почте могут пересылаться самые различные текстовые файлы, изображения, программы и наборы данных.

**Текст письма** для отправки его по электронной почте должен быть подготовлен на компьютере в виде текстового файла и оформлен в соответствии со стандартами Интернет. Общепринятый формат по­слания состоит из заголовка и непосредственно сообщения, роль которого может играть любой текстовый файл, рисунок или про­грамма.

Стандартный заголовок электронного письма выглядит следую­щим образом:

**From:** *адрес отправителя*

**То:** *адрес получателя*

**Сс:** *другие адреса отправки*

**Subject:** *тема сообщения*

Пример почтового адреса в сети Интерент:

[vitkay@mail.ru](mailto:vitkay@mail.ru)

Первое имя **vitkay** - это идентификатор владельца электронного почтового ящика. Второе имя - адрес почтового сервера **mail.ru.** Здесь **mail** - идентификатор отечественной почтовой системы Mail-Ru, в которой любой из Вас может открыть себе бесплатный электронный почтовый ящик.

Для поиска информации в сети Интернет в нашей стране и за рубежом используется несколько информационно-поисковых сис­тем. Среди отечественных систем наиболее известны системы Апорт, Ремблер и Яндекс, зарубежных - Altavista, Infoseek, Yahoo.

Поиск информации в сети Интернет этими системами произ­водится по запросам. Простейшие запросы состоят из одного или нескольких слов на русском или английском языках, либо на другом языке, на котором записаны искомые документы.

Результатом поиска в сети Интернет являются перечни названий и адресов гипертекстов, отвечающих заданным запросам. Например, на запрос «Кайман информатика» поисковые системы предоставят список всех гипертекстов, доступных в Интернете и в которых ука­заны слова «Каймин» и «информатика». В том числе в этом списке будут указаны адреса серверов, на которых размещены сетевые электронные учебники по информатике.

Указанные поисковые системы еженедельно просматривают все серверы в сети Интернет и индексируют все найденные гипертексты, запоминая их адреса и встречающиеся в них ключевые слова и сло­восочетания. По этой причине любая информация, выставленная в сети Интернет, может быть найдена по интересующим нас запросам.

**В о п р о с ы**

1. Что такое Интернет?

2. Что такое вычислительная сеть?

3. Какими бывают вычислительные сети?

4. Что такое сервер?

5. Что такое - информационные ресурсы?

6. Что такое WWW?

7. Что такое гипертекст?

8. Что такое электронная почта?

9. Как образуются адреса в сети Интернет?

10. Как ищется информация в сети Интернет?

**З а д а н и я**

1. Найдите в сети Интернет сервер радиостанции **«Русское** **радио**».

2. Найдите в сети Интернет сервер газеты **Известия.**

3. Найдите в сети Интернет игровой сервер.

4. Найдите в сети Интернет вузы, занимающиеся дистанционным обучением.

5. Найдите в сети Интернет серверы с рефератами по истории и литературе.

6. Откройте себе почтовый ящик в Интернет.

## 2.2. Базы данных на ЭВМ

Одним из видов информационных технологии на основе ЭВМ являются базы данных. В отличии от обычных файлов базы данных допускают определенные процедуры поиска и выборки информации, хранящейся в памяти вычислительных машин.

**База данных** - это совокупность данных, хранящихся в долговре­менной памяти ЭВМ и допускающих определенные способы поиска информации. В форме баз данных могут храниться различные сведе­ния: расписание движения поездов, автобусов и самолетов, сведения о наличии товаров в магазине или на складе, сведения о студентах, преподавателях и сотрудниках, информация о книгах и многое, многое другое.

Информация в базах данных может быть организована несколь­кими способами. **Табличная форма** - наиболее простая и распрост­раненная форма организации баз данных, получившая название реляционной. Пример таблицы:

**Фамилия Имя Рост Вес Глаза**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Иванов | Саша | 180 | 85 | синие |
| Петрова | Оля | 165 | 65 | карие |
| Сидоров | Миша | 190 | 75 | зеленые |

**Реляционные базы** - это базы данных, в которых информация хранится в форме таблиц. Каждая из таблиц в такой базе данных имеет строго определенное число столбцов, снабженных именами.

В рассматриваемом примере таблица имеет пять столбцов с име­нами **фамилия, имя, рост, вес** и **глаза.** В каждом из столбцов могут записываться данные определенного типа - числового или символь­ного. Так в столбцах **вес** и **рост** записываются числовые данные, а в столбцах **фамилия, имя** и **глаза** записывается символьная информа­ция.

**Данные об объектах,** людях или вещах в этих таблицах записыва­ются в виде строк. В приведенном примере сведения о росте, весе и цвете глаз Петровой Оли записаны во второй строке.

**Порядок строк** в таблицах задается при вводе данных. Однако во многих базах данных имеется возможность сортировки данных в таблицах. Наиболее распространенным видом сортировок в базах данных является упорядочение информации по одному из столбцов.

Для числовых данных **упорядочение** проводится по возрастанию или убыванию значений. Например, упорядочение по росту:

**Фамилия Имя Рост Вес Глаза**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Петрова | Оля | 165 | 65 | карие |
| Иванов | Саша | 180 | 85 | синие |
| Сидоров | Миша | 190 | 75 | зеленые |

Упорядочение символьных данных состоит в расположении их алфавитном порядке. Пример упорядочения по именам:

**Фамилия Имя Рост Вес Глаза**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сидоров | Миша | 190 | 75 | зеленые |
| Петрова | Оля | 165 | 65 | карие |
| Иванов | Саша | 180 | 85 | синие |

Наиболее важной возможностью баз данных является автомати­ческий поиск информации. **Поиск данных** - это отбор данных по четко определенной комбинации признаков.

Основой для поиска информации в базах данных служат **запросы.** Совокупность запросов, по которым можно получить информацию, считается главной характеристикой баз данных.

**Запросы** в базах данных подразделяются на простые и сложносоставные. В простых запросах указывается имя одного из столбцов и некоторое значение. Примеры простых запросов:

запрос: **фамилия =** Иванов

запрос: **имя** = Оля

**Ответами на запросы** будут строки из таблицы приведенного типа. На первый запрос - строки, в которых в графе **фамилия** стоит «Иванов», а на второй запрос - строки со значением «Оля» в графе **имя.**

Для числовых значений в запросах могут стоять знаки «больше» или «меньше». Примеры соответствующих запросов:

запрос: **рост** > 180

запрос: **вес ≤** 50

Ответами на эти запросы также будут одна или несколько строк из таблицы, в которых **рост** или **вес** будут удовлетворять указанным условиям. Но возможно, что поиск окажется безрезультативным, если ни одна строка в таблице не удовлетворяет заданным условиям.

**Сложносоставные запросы** образуются из простейших с исполь­зованием логических связок **и** и **или.** Примеры сложносоставных запросов:

запрос: **вес <** 80 **и глаза** = зеленые

запрос: **глаза** = синие **или глаза** = голубые

Ответами на составные запросы со связкой и будут все строки таблицы, удовлетворяющие обоим условиям. Ответами на запросы со связкой **или** будут все строки таблицы, которые удовлетворяют первому или второму условию, либо и тому и другому одновременно.

Отличие баз данных от **информационно-справочных** и информа­ционно-поисковых систем состоит в следующем. В информаци­онно-справочных системах допустим только поиск информации. А в базах данных можно выполнять не только поиск информации, но и обновлять их, а также создавать новые таблицы либо реоргани­зовать их.

Обновление данных в таблицах баз данных проводится по эле­ментам строк, а добавление информации - вводом новых строк. Для создания новой таблицы необходимо задать ее имя, число столбцов и указать типы значений в каждом из столбцов.

**Задача 1.** База данных об оценках.

Составьте базу данных об оценках своих товарищей, выделив следующие предметы: математика, физика и информатика. Укажите фамилии товарищей, их имена и оценки по этим предметам. Приве­дите примеры простых и сложносоставных запросов.

Р е ш е н и е. Пусть имеются три товарища: Иванов, Петрова и Сидоров со следующими оценками по физике, математике и инфор­матике:

**фамилия имя матем физика информ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Иванов | Саша | 5 | 4 | 5 |
| Петрова | Оля | 4 | 4 | 5 |
| Сидоров | Миша | 3 | 3 | 4 |

Примеры запросов:

**фамилия** = Петрова

**имя** = Миша

**физика > 3**

**матем** > 3 **и** **физика** > 3

**матем** = 5 **или** **информ** = 5

**В о п р о с ы**

1. Что такое база данных?

2. Что такое реляционные базы данных?

3. Что такое сортировка данных?

4. Как упорядочивается информация в базах данных?

5. Что такое запросы к базам данных?

6. Как строятся сложносоставные запросы?

7. Каковы основные возможности баз данных?

**З а д а н и я**

1. Составьте базу данных о кондитерских товарах, указав их назва­ние, вес, цену и вкус. Заполните базу данных на 5-6 наименований конфет. Приведите примеры сложно-составных и простых запросов с нетривиальными ответами.

2. Составьте базу данных о своих родных: маме, папе, сестрах, братьях, дедушках и бабушках с указанием их дней рождения и месте работы или учебы. Упорядочите базу данных по возрасту и приведите примеры запросов.

3. Составьте базу данных о своих друзьях с указанием их возраста, места учебы, профессий и любимых увлечений. Упорядочите базу дан­ных в алфавитном порядке по именам друзей и приведите примеры запросов.

4. Составьте базу данных о своих учителях с указанием фамилий, имен, отчеств, возраста и предмета преподавания. Упорядочите базу данных в алфавитном порядке по фамилиям преподавателей и приведите примеры запросов.

5. Составьте базу данных по своей успеваемости с указанием оценок по литературе, физкультуре, математике, физике и информатике. Упорядочите базу данных в порядке убывания оценок по: а) литературе, б) физкультуре, в) математике.

6. Составьте базу данных по лучшим спортсменам года по любимому виду спорта с указанием лучших результатов или мест на ведущих со­ревнованиях.

7. Составьте по журналу успеваемости базу данных по следующим предметам: а) математике; б) информатике; в) физике; г) литературе.

Укажите запросы на поиск учеников, не имеющих

а) ни одной двойки; в) ни одной тройки;

б) ни одной четверки; г) ни одной пятерки.

8. Составьте базу данных «Телефонный справочник» с телефонами своих друзей и родных с указанием фамилий и имен. Упорядочите базу данных по фамилиям.

## 2.3. Выполнение расчетов на компьютерах

Персональные компьютеры служат удобным **средством вычис­лений** и **расчетов** экономического и математического содержания. В этом смысле компьютеры намного эффективнее бухгалтерских счетов и калькуляторов, которые требуют больших затрат ручного труда.

Наиболее удобным средством проведения расчетов на персональ­ных компьютерах являются **электронные таблицы.** В этих програм­мах все исходные и расчетные данные отображаются на экране в форме таблиц.

**Электронные таблицы** - программы для выполнения и хранения различных расчетов и калькуляций на компьютерах. На персональ­ных компьютерах **IBM PC** наибольшее распространение получили электронные таблицы **Excel.**

**Калькуляция** - это таблица с определенным набором надписей, формул и данных, используемых для расчетов. В качестве примера рассмотрим калькуляцию закупки сладостей к дню рождения.

Пусть к дню рождения принято решение купить шоколад и кон­феты «Аленка», «Мишки», «Марс». Соответствующая калькуляция закупок конфет с учетом их цен, веса и количества имеет вид:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** |
| **1** | конфеты | цена | вес | кол-во | масса | стоим |
| **2** | Аленка | 600 | 100 | 3 | 300 | 1800 |
| **3** | Мишки | 200 | 40 | 10 | 400 | 2000 |
| **4** | Марс | 500 | 100 | 5 | 500 | 2500 |
| **5** |  |  |  |  |  |  |
| **6** |  |  | итого: | 18 | сумма: | 6300 |
| **7** | денег: | 8000 |  |  | остаток: | 1700 |

С помощью приведенной калькуляции, меняя на экране коли­чество конфет, можно оценить различные варианты закупок. В част­ности, можно решить следующие проблемы:

купить на заданную сумму наибольшее число сладостей;

купить на заданную сумму наибольшую массу конфет;

купить все виды конфет для пяти гостей в рамках заданной суммы.

Р е ш е н и е. Для выполнения расчетов в электронной таблице должна храниться следующая система формул:

Е2 = C2\*D2 F2 = B2\*D2

ЕЗ = C3\*D3 F3 = B3\*D3

E4 = C4\*D4 F4 = B4\*D4

D6 = D2 + D3 + D4 F7 = B7 - F6

F6 = Sum (F2 : F4)

Здесь B2, D3, E6, F7 - имена ячеек электронной таблицы; Sum (F2 : F4) - функция суммирования ячеек из столбца F от ячей­ки F2 до ячейки F4.

**Основные возможности** электронных таблиц на компьютерах:

1) автоматический перерасчет калькуляций;

2) хранение и поиск калькуляций в памяти ЭВМ;

3) вывод калькуляций на печать;

4) обновление и ввод новых калькуляций.

**Перерасчет калькуляций** в электронных таблицах производится автоматически сразу же после обновления на экране любых исход­ных данных. В этом заключается основное свойство и удобство эле­ктронных таблиц: один раз составленная калькуляция может исполь­зоваться многократно для выполнения расчетов при различных ис­ходных данных.

**Хранение** калькуляций в электронных таблицах обычно проводится на магнитных дисках. Это позволяет повторно использовать их для новых расчетов и перерасчетов. Бумажная копия любой из электрон­ных таблиц со всеми ее исходными и расчетными данными может быть выведена на печать.

**Ввод калькуляций,** состоящих из надписей, числовых данных и формул, проводится по ячейкам. Для этого к необходимой ячейке подводится курсор с помощью мышки или клавиш стрелок, а затем нажимается клавиша Enter на клавиатуре либо клавиша на мышке.

**Копирование и перенос** надписей, данных, формул и целых блоков таблиц позволяет достаточно быстро создавать новые калькуляции из уже имеющихся в памяти компьютера. Многие электронные таб­лицы допускают изменение размеров строк или столбцов таблиц для их более наглядного и красивого расположения.

**Числовые данные** могут быть целыми и вещественный числами. Примеры записи чисел в электронных таблицах:

0, 1, 2, 3, ... , -1, -2, -3, ... - целые числа;

0.1, 1.5, 12.87, 0.002 , ... - вещественные числа.

Обратите внимание: для записи дробной части обычно приме­няется точка, а не запятая. Для записи десятичного порядка исполь­зуется символ Е:

1.2Е6 ≡ 1200000

-.5Е-4 ≡ -0.0005

**Расчетные формулы** в электронных таблицах образуются из чис­ловых значений, обозначений элементарных и специальных функ­ций и имен ячеек электронной таблицы: А1, А2, A3, В1, В2, С1 и т. д.

Запись **арифметических операций** в формулах и числовых выраже­ниях в электронных таблицах выполняется с помощью следующих знаков:

+ - сложение 2+2 А2+В2+С2

- - вычитание 6-8 А1-В1

\* - умножение 7\*8 2\*А2\*С2

/ -деление 2/3 А1/(2/С2)

∧ - возведение в степень 5∧3 A3∧2

**Математические функции** в электронных таблицах имеют следую­щие обозначения:

sin(x) - синус cos(x) - косинус

tan(x) - тангенс atan(x) - арктангенс

ехр(х) - экспонента ln(x) - натуральный логарифм

sqr(x) - квадратный корень

**В о п р о с ы**

1. Что такое калькуляция?

2. Каковы основные возможности электронных таблиц?

3. Какие электронные таблицы используются на IBM PC?

4. Как записываются формулы в электронных таблицах?

5. Какие математические функции есть в электронных таблицах?

**З а д а н и я**

1. Составьте систему формул для расчета заработной платы по следу­ющей таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **1** | фамилия | часы | оплата | з/плата |
| **2** | Иванов | 20 | 1000 | 20000 |
| **3** | Петрова | 25 | 800 | 20000 |
| **4** | Сидоров | 10 | 600 | 6000 |
| **5** |  |  |  |  |
| **6** |  |  | итого: | 46000 |

2. Составьте калькуляцию для закупок письменных принадлежнос­тей:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** |
| **1** | Закупки: | цена | колич | сумма |  |  |
| **2** | тетради | 200 | 10 | 2000 |  |  |
| **3** | карандаши | 300 | 8 | 2400 |  |  |
| **4** | ручки | 3500 | 4 | 14000 |  |  |
| **5** | ластики | 400 | 2 | 800 |  |  |
| **6** |  |  |  |  |  |  |
| **7** |  |  | всего: | 19200 |  |  |
| **8** |  |  |  |  |  |  |

3.Составьте калькуляцию закупок продуктов для похода на N дней и М человек.

## 2.4. Постановка и решение задач

**Решение задач** состоит в получении определенных результатов. Это относится к в работе, жизни или учебе: сдача экзаменов, написание сочинений, выполнение чертежей, изготовление приборов, инстру­ментов и машин, сбор урожая, накопление капитала и т. п. - все это получение или достижение результатов.

Ключом к любой задаче является **способ решения,** дающий необ­ходимые результаты. Знание способов решения и умение их приме­нять для решения практических задач - важнейшая характеристика профессиональной квалификации.

**Результаты правильные,** если они отвечают требованиям решае­мых задач. Однако, если требования сформулированы недостаточно четко, то нельзя однозначно судить о правильности полученных ре­зультатов.

**Результаты неправильные,** если они противоречат заданным требованиям. Как однозначно определить правильность результатов? Ответ: для этого необходима точная постановка задач с четким выделением требований.

Для решения задач необходимо определение:

1) **что требуется?**

2) **что дано?**

Ответ на первый вопрос **- что требуется? -** точное определение требуемых результатов. При отсутствии требований к конечным целям оценка полученных результатов может быть неоднозначной.

Ответ на второй вопрос - **что дано?** - определение исходных условий, при которых требуется получить результаты. Неоднознач­ность в определении исходных условий может привести к получе­нию неправильных результатов.

Рассмотрим задачу: **«Добраться домой».** Исходным будет место, где мы находимся, а требуемым - свой дом. Способов решения этой задачи может быть много, но правильные среди них только те, кото­рые обеспечат достижение своего дома.

Рассмотрим вторую задачу. «Решение уравнения **2⋅х+1** ***=* 0**». Здесь требуемым является корень уравнения. В качестве решения уравне­ния можно рассмотреть два числа х1 = 1 и х2 = -1/2. Правильным из них является то решение, при подстановке которого уравнение пре­вратится в тождество.

Подстановка первого числа х1 = 1 в уравнение дает противоречие

2.(1) +1= 3 ≠ 0.

Следовательно, значение х1 = 1 - это неправильное решение, так как оно противоречит требованиям и не может быть корнем уравне­ния.

Подстановка второго решения х2 = -1/2 в уравнение дает тожде­ство

2.(-1/2) +1= 0.

Таким образом значение х2 = -1/2 удовлетворяет исходному урав­нению и является правильным решением.

**Способ решения правильный,** если он дает правильные результаты. Для определения правильности способов решения задач необходима четкая постановка решаемых задач, в которых должны быть строго определены требуемые результаты.

**Способ - неправильный,** если его применение приводит к получе­нию неправильных результатов либо вовсе не дает никаких резуль­татов. Использование неправильных способов решения может вооб­ще не давать результатов.

Способы могут быть частными и общими. **Частные способы** дают конкретные решения частных задач. Частный способ может оказать­ся неприменимым для решения сходных задач, отличающихся дета­лями.

**Общий способ** может давать решения для целого класса задач, отвечающих определенным исходным условиям и отличающихся друг от друга конкретными исходными данными.

Так, для рассмотренной задачи решения уравнения 2-х + 1 = 0 можно использовать общий способ решения линейных уравнений вида **а⋅х + b = 0:**

х0 = - b/а.

Применение этой формулы при а = 2, b = 1 дает решение х0 = - b/а = -1/2, которое нам уже известно как правильное.

В правильности общего способа решения уравнений вида а⋅х + b = 0 можно убедиться подстановкой формулы х0 = - b/а в само уравне­ние:

а⋅х + b ≡ а⋅(- b/а) + b ≡ -b + b ≡ 0.

При **постановке обобщенных задач** кроме выделения требуемого необходимо определить исходные условия, при которых должно быть получено требуемое. В такой постановке задач должно быть опреде­лено, какие исходные условия будут считаться допустимыми, а ка­кие нет.

**Постановка задачи:**

1. Что дано?

2. Что требуется?

3. Что допустимо?

Приведем полное описание постановки рассмотренной выше за­дачи:

**Задача:** решить уравнение а-х + b *=* 0.

Треб: х - корень уравнения.

Дано: а, b - коэффициенты уравнения.

При: а ≠ 0.

Уравнения данного типа можно решать в общем виде с помощью электронных таблиц, применяя описанный общий метод решения и следующую калькуляцию:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **1** |  | уравнение: |  |  |
| **2** | 2 | \* х + | 1 | = 0 |
| **3** | корень: | х = -0.5 |  |  |

с расчетной формулой С3 = -С2/ А2.

Особую ценность для решения задач представляют обобщенные методы решения. **Метод** - единый способ решения некоторого класса задач. Знание методов позволяет находить решения для любой кон­кретной задачи данного класса.

**Метод решения правильный,** если он дает правильные результаты для любой задачи данного класса. Применение таких методов гаран­тирует правильность результатов для любой из задач данного класса.

**Метод решения неправильный,** если можно указать конкретную задачу данного класса, для которой применение метода даст непра­вильные результаты либо не даст результатов вовсе.

Например, для уравнения а⋅х + b = 0 формула х = - b/а не дает результата при а = 0. Но при значении а = 0 уравнение превращается в соотношение b = 0, что говорит о недопустимости этого значения. Следовательно, условием допустимости данных в рассматриваемой задаче будут значения а ≠ 0.

Правильность методов решения можно проверять на конкретных примерах. Достаточно привести хотя бы один **контрпример,** на кото­ром способ или метод дает неправильный результат, чтобы утверж­дать о неправильности метода решения в целом.

Однако **демонстрация правильности** результатов на двух-трех при­мерах не может служить достаточным основанием для утверждений о правильности метода или способа решения в целом.

**Полное обоснование** правильности методов решения дает только исчерпывающий анализ результатов, получаемых с их помощью для любых задач данного класса. Пример - приведенное выше обосно­вание общего метода решения линейных уравнений.

В общем случае **обоснование правильности** обобщенных методов решения требует математического исследования получаемых резуль­татов и математического доказательства их правильности для всех конкретных случаев.

**В о п р о с ы**

1. Когда результаты правильные?

2. Когда результаты неправильные?

3. Когда способ решения правильный?

4. Что такое постановка задачи?

4. Что такое метод решения?

5. Когда метод решения правильный?

6. Когда метод решения неправильный?

**З а д а н и я**

1. Приведите постановку задачи и общий метод решения квадрат­ного уравнения.

2. Приведите калькуляцию для решения квадратных уравнений на компьютере.

3. Докажите правильность общего метода решения квадратного урав­нения.

4. Приведите калькуляцию для решения системы уравнений с двумя неизвестными:

а⋅х + b⋅у = е

с⋅х + d⋅y = f

с помощью следующего общего метода:

х = Dx/D у = Dy/D

Dx = e⋅d - b⋅f Dy = a⋅f - b⋅e

D = a⋅d - b⋅c

# Глава 3. ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

## 3.1. Элементы математической логики

Понятие **«искусственный интеллект»** возникло с появлением самых первых компьютерных программ, имитирующих интеллектуальную деятельность людей - игру в шахматы, шашки, доказательство теорем и решение задач на ЭВМ.

Все компьютерные программы, демонстрирующие **интеллектуальное поведение,** основаны на использовании определенного математичес­кого аппарата, опирающегося на законы математической логики. Без понимания этих законов невозможно понимание принципов работы вычислительных машин вообще и систем искусственного интеллекта в частности.

**Логика** - это наука, изучающая правильность суждений, рассуж­дений и доказательств. Примеры суждений: *«снег белый»,* «2×2 *= 5», «Земля круглая», «информатика - наука», «генетика - лженаука».*

**Суждения** могут быть истинными или ложными. Истинность или ложность суждений проверяется их соответствием действительности. Пример истинного суждения - *«снег белый».* Пример ложного суж­дения - *«генетика - лженаука».*

**Суждение истинно,** если оно отражает действительное положение вещей. Примеры истинных суждений: *«снег белый»,* «2×2 = 4», *«театр - это искусство».*

**Суждение ложно,** если оно противоречит истинному положению вещей. Примеры ложных утверждений - «2×2 = 5», *«снег - черный», «Земля плоская».*

Однако существуют суждения, об истинности или ложности которых нельзя судить однозначно. Пример таких суждений: *«есть жизнь на Марсе», «машина может думать», «астрология - наука».*

**Математическая логика** - это дисциплина, изучающая технику математических доказательств. Отличие математических суждений от обычных разговорных высказываний состоит в том, что математи­ческие суждения всегда предполагают однозначную интерпретацию, в то время как наши обычные высказывания зачастую допускают многозначную трактовку.

**Математика** - наука, признающая исключительно только одно­значные суждения, утверждения и допускающая только строгие до­казательства. В то время как люди в своих рассуждениях и высказы­ваниях допускают различного рода неточности и двусмысленности.

**Работа ЭВМ** как автоматических устройств основана исключи­тельно на математически строгих правилах выполнения команд, программ и интерпретации данных. Тем самым работа компьютеров допускает строгую однозначную проверку правильности своей работы в плане заложенных в них процедур и алгоритмов обработки инфор­мации.

**Фундаментом науки** о вычислительных машинах является конст­руктивная математика, в основе которой лежит математическая ло­гика и теория алгоритмов с их однозначностью в оценке суждений и процедур вывода. Математическая логика с самого начала использо­валась для описания элементов и узлов ЭВМ, а теория алгоритмов - для описания компьютерных программ.

Основными объектами в математической логике являются **- высказывания и предикаты.** Первые изучаются в исчислении выска­зываний, а вторые - в исчислении предикатов.

**Высказывания** - это суждения, о которых может быть известно - что они истины или ложны. В исчислении высказываний не иссле­дуется - о чем утверждается в этих суждениях.

Высказывания обычно обозначаются отдельными буквами или буквами с возможными индексами. Примеры простых высказываний и их обозначений:

**А** = «снег белый»

**В1** = «вода теплая»

**В2** = «земля твердая»

С математической точки зрения высказывания - это перемен­ные, принимающие значения **«истина»** или **«ложь».** Эти два истин­ностных значения иногда заменяются словами «да», «нет», либо цифрами 1 и 0.

В отличии от высказываний **предикаты** - это суждения о некото­рых переменных объектах или их свойствах. Примеры предикатов:

**А(х)** = *«цвет яблока -* **х**»

**В(х,** **у)** = «**х** < **у**»

где х, у - это некоторые переменные (объекты).

**Значениями переменных** в предикатах могут быть числа, слова, вектора, списки, функции, процедуры, алгоритмы или даже про­граммы. Для математической логики существенно, чтобы эти пере­менные объекты имели конструктивную форму и были бы строго определены.

С математической точки зрения **предикаты - это функции,** име­ющие одну или несколько переменных и принимающие логические значения «истина» или «ложь». Обозначения предикатов в матема­тической логике схожи с обозначениями обычных математических функций: **Р(х), Q(x,y)** и т. д.

В информатике для **обозначения переменных, функций и предика­тов,** а также их аргументов обычно используются осмысленные сло­ва и словосочетания в целях простоты их ввода в ЭВМ. Например, предикаты, используемые для описания фактов в языке Пролог, обыч­но имеют обозначения, выражаемые в лексике родного языка:

**любит (Маша, х);**

**цена (конфеты, с).**

В **форме предикатов** с конкретными аргументами-значениями могут быть описаны факты любой базы данных. Примеры описания фактов из базы данных в записи на языке Пролог:

**любит (Маша, цветы)** - *Маша любит цветы*

**любит (Саша, машины)** - *Саша любит машины*

**цена (цветы, 1000)** - *цена цветов 1000*

**цена (мороженое, 2500)** - *цена морженого 2500*

В этой же форме предикатов с переменными могут описываться и **простейшие запросы** к базам данных на языке Пролог. Примеры запросов к указанной базе данных на языке Пролог и соответствую­щие ответы ЭВМ:

**? любит** **(х, конфеты)** - *Кто любит конфеты?*

**х** = **Маша**

**? цена (конфеты, с)** - *Какова цена конфет?*

**с = 1000**

**В о п р о с ы**

1. Что изучает математическая логика?

2. Что изучает логика?

3. Что такое высказывание?

4. Что такое предикат?

5. Когда суждения истинны?

6. Когда суждения ложны?

**З а д а ч и**

1. Приведите примеры истинных и ложных утверждений

а) из арифметики;

б) из геометрии;

в) из биологии;

г) из жизни.

2. Выразите отрицания для высказываний:

а) *«мы пойдем в кино»;*

б) «х = 0 **или** х = 1»;

в) «х = 0 **и** у = 0»;

г) «а = 0 **и** b = 0 **и** с = 0»;

д) «х = 0 **или** у = 0 **или** z = 0».

е) *«мы не пойдем никуда»;*

ж) «а = 0 **или** b = 0»*;*

з) «х > 0 **и** х < 100».

## 3.2. Основные логические операции

**Суждения** в математической логике могут быть простыми и сложносоставными. Примеры простых суждений:

**х = 1 рост < 160**

**А цена (х, у)**

**Сложносоставные суждения** в математической логике образуются из простых с помощью логических связок **и, или** и **не,** выражающих три основных логических операции:

логическая связка **не** - **отрицание** суждений;

логическая связка **или - конъюнкция** суждений;

логическая связка **и** - **дизъюнкция** суждений.

Примеры сложносоставных суждений:

**не А** - *неверно суждение* **А**

**С или В** - *истинно***С или В**

(х > 0) **и** (у > 0) - (**х** *больше* 0) **и** (**у** *больше* 0)

(**глаза** = синие) **или (глаза** = голубые)

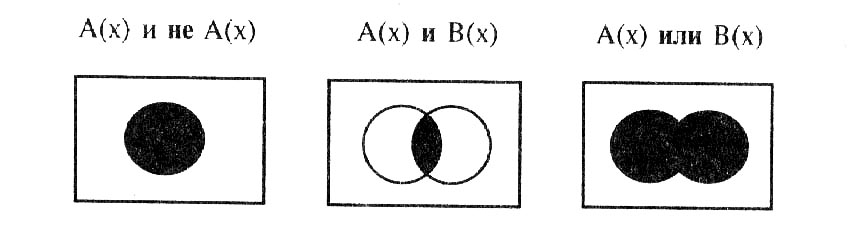
Логическая связка не используется для выражения отрицаний. Примеры:

**не (глаза** = синие), - *неверно, что глаза синие*

**не** (А **или** В), - *неверно, что выполняется А или В*

**не** (любит (Саша, конфеты)) - *неверно, что Саша любит конфеты*

Наглядной иллюстрацией этих логических связок с предикатами служат следующие диаграммы:



Отрицание **не А** истинно или ложно в зависимости от истинности исходного суждения **А**. Свойства отрицания **не** как логической связки можно описать таблицей истинности:

Таблица истинности:

**А не А**

|  |  |
| --- | --- |
| **да** | **нет** |
| **нет** | **да** |

Свойства отрицаний:

НЕ1: **Отрицание ложно,** если суждение истинно.

НЕ2: **Отрицание истинно,** если суждение ложно.

**Для понимания отрицаний** важно уметь выражать их в позитивной форме. Приведем примеры отрицания математических неравенств и их позитивные переформулировки:

**не** (х = 0) ≡ (х ≠ 0)

**не** (х ≠ 0) ≡ (х = 0)

**не** (х > 0) ≡ (х ≤ 0)

**не** (х < 0) ≡ (х ≥ 0)

**не** (х ≥ 0) ≡ (х < 0)

**не** (х ≤ 0) ≡ (х > 0)

Свойства отрицаний, записанные в таблицу истинностности, могут быть описаны как факты на языке Пролог:

**не (да, нет);**

**не (нет, да);**

После ввода этих фактов в ЭВМ с помощью запросов можно перепроверить свойства отрицаний:

**? не** **(А, нет)**

**А = да**

**? не (А, да)**

**А = нет**

Логическая связка **и** в математической логике называется конъ­юнкцией. Таблица истинности **конъюнкции:**

**А В А и В**

|  |  |  |
| --- | --- | --- |
| **да** | **да** | **да** |
| **да** | **нет** | **нет** |
| **нет** | **да** | **нет** |
| **нет** | **нет** | **нет** |

Свойства конъюнкции:

И1: **Конъюнкция** А и В **истинна,** когда истинны оба суждения.

И 2: **Конъюнкция** А и В **ложна,** когда ложно хотя бы одно из суж­дений А или В.

Логическая связка **или** в математической логике называется дизъ­юнкцией. Таблица истинности **дизъюнкции:**

**А В А или В**

|  |  |  |
| --- | --- | --- |
| **да** | **да** | **да** |
| **да** | **нет** | **да** |
| **нет** | **да** | **да** |
| **нет** | **нет** | **нет** |

Свойства дизъюнкции:

ИЛИ1: **Дизъюнкция** А или В **истинна,** когда истинно любое из суждений А или В.

ИЛИ2: **Дизъюнкция** А или В **ложна,** когда ложны оба суждения А и В.

Свойства конъюнкции и дизъюнкции также можно описать в виде фактов на языке Пролог:

Дизъюнкция: Конъюнкция:

**или (да, да, да); и2 (да, да, да);**

**или (да, нет, да); и2 (да, нет, нет);**

**или (нет, да, да); и2 (нет, да, нет);**

**или (нет, нет, нет); и2 (нет, нет, нет);**

Опираясь на эти факты можно получить ответы на вопросы о свойствах дизъюнкции и конъюнкции с помощью ЭВМ:

**? или (А, В, нет) ? и 2 (А, В, да)**

**А = нет В = нет А = да В = да**

**? или (А, В, да) ? и 2 (А, В, нет)**

**А = да В = да А = да В = нет**

**А = да В = нет А = нет В = да**

**А = нет В = да А = нет В = нет**

Одной из важнейших логических связок математической логики является импликация **А → В**. Эта связка в математической логике используется для определения правил логического вывода.

**Импликация** **А →** **В** - это логическое следование. Импликация **А → В** читается: «если А, то В». Первое суждение в импликации называется посылкой, а второе суждение - следствием.

Приведем примеры правил логического вывода:

а) с использованием высказываний:

если *«на улице дождь»,* то *«на улице мокро»,*

б) с использованием предикатов:

**любит (х, конфеты) → сластена (х).**

Таблица истинности импликации:

**А В А → В**

|  |  |  |
| --- | --- | --- |
| **да** | **да** | **да** |
| **да** | **нет** | **нет** |
| **нет** | **да** | **да** |
| **нет** | **нет** | **да** |

Свойства импликации:

П1: «**Импликация** А → В **ложна**,

когда посылка А истинна, а следствие В - ложно».

П2: «**Импликация** А → В **истинна**,

когда истинно следствие либо ложны и посылка и следствие».

В языке Пролог импликации используются для описания правил вывода и определения новых логических понятий. Например, поня­тие «сластена» в языке .Пролог описывается следующим образом:

**сластена (х) ← любит (х, конфеты);**

Описание этого правила позволяет вводить в ЭВМ вопросы о «сластенах» и получать осмысленные ответы, исходя из сведений, хранящихся в базе данных:

**? сластена (х)** - *Кто сластена?*

**х = Маша**

С помощью таблиц истинности могут быть описаны и проверены свойства любых сложносоставных высказываний. Соответственно с помощью этих таблиц на ЭВМ средствами языка Пролог могут быть проверены любые сложносоставные высказывания и законы исчис­ления высказываний.

**Задача 1.** Проверьте закон двойного отрицания в исчислении высказываний

**не** (**не** А) ≡ А

Р е ш е н и е *.* Рассмотрим объединенную таблицу истинности вы­сказываний

**А не А не (неА)**

|  |  |  |
| --- | --- | --- |
| **да** | **нет** | **да** |
| **нет** | **да** | **нет** |

Сравнение крайних столбцов показывает, что всюду, где выска­зывание А истинно, там же истинно и двойное отрицание **не** (**не** А). И наоборот, всюду, где ложно А, там ложно и двойное отрицание **не** (**не** А). Следовательно, двойное отрицание тождественно исходному высказыванию: **не** (**не** А) ≡ А.

**Задача 2.** Сравните с помощью таблиц истинности отрицание дизъюнкции и отрицание конъюнкции **не** (А **и** В) и **не** (А **или** В).

Р е ш е н и е .

АВА **и** В **не (**А **и** В**)** А **или** В **не (**А **или** В**)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| да | да | да | нет | да | нет |
| да | нет | нет | да | да | нет |
| нет | да | нет | да | да | нет |
| нет | нет | нет | да | нет | да |

**В о п р о с ы**

1. Когда истинно отрицание?

2. Когда ложна дизъюнкция?

3. Когда истинна конъюнкция?

4. Когда ложна импликация?

**З а д а н и е**

1. Составьте таблицы истинности для утверждений:

а) (**не** А) **и** (**не** В); в) (**не** А) **или** (**не** В);

б) А **и** (**не** В); г) А **или** (**не** В).

2. Сравните с помощью таблиц истинности логические выражения:

а) **не** (А **и** В); в) (**не** А) **или** (**не** В);

б) **не** (А **и** В); г) (**не** А) **или** (**не** В).

3. Проверьте по таблицам истинности логические законы:

а) отрицание конъюнкции:

**не** (А **и** В) = (**не** А) **или** (**не** В);

б) отрицание дизъюнкции:

**не** (А **или** В) = (**не** А) **и** (**не** В);

в) отрицание импликации:

**не** (А → В) ≡ (**не** В) → (**не** А).

## 3.3. Элементы языка Пролог

**Пролог** - это одна из моделей систем искусственного интеллекта, способных воспроизводить логические умозаключения. Кроме того, Пролог - это язык для описания фактов, правил и процедур логи­ческого вывода. О языке Пролог обычно говорят, что он представляет язык логического программирования.

Основной особенностью системы Пролог являются встроенные **процедуры логического вывода**, имитирующие способность человечес­кого интеллекта выполнять логические умозаключения. Какая лек­сика используется при этом - русского, английского или других языков - играет второстепенную роль, несущественную для ЭВМ, но важную для тех, кто будет работать и вести диалог с машиной.

Основная идея Пролога **как языка записи** фактов, вопросов и правил заключается в том, что они записываются в форме предика­тов математической логики. Все они интерпретируются ЭВМ строго в соответствии с законами математической логики и ни чем более.

**Основные конструкции** языка Пролог - это факты, вопросы и правила. Все эти три конструкции записываются в форме преди­катов и их комбинаций. Рассмотрим правила их записи на языке Пролог.

**Факты** - это конкретные сведения о ком-то либо о чем-то. Факты на языке Пролог записываются в форме предикатов с конкретными аргументами-значениями. Примеры записи фактов на Прологе:

**папа (Вова, Лена);** - *Вова - папа Лены*

**любит (Лена, музыка);** - *Лена любит музыку*

**оценка (Лена, русский, 5);** - *У Лены 5 по русскому языку*

**Вопросы** на Прологе - это запросы к совокупности данных или процедурам, хранящимся, в ЭВМ. Запись вопросов начинается со знака **?**, за которым записывается предикат или группа предикатов, разделяемых запятыми. Примеры записи простых вопросов на языке Пролог:

**? папа (х, Лена)** - *Кто папы Лены?*

**х = Вова**

**? мама (х, у)** - *Кто у кого - мама ?*

**НЕТ**

**? оценка (х, \_ , 5)** - *Кто имеет оценки 5?*

**х = Лена**

Здесь буквы **х**, **у** - обозначения переменных, а числа и слова - конкретные значения аргументов в соответствующих предикатах. Знак подчеркивания «**\_**» представляет неопределенное значение, которое несущественно для ответа на вопросы.

При записи **сложносоставных вопросов** в языке Пролог можно указывать несколько условий-предикатов, разделяемых запятыми. Запятая в этих сложносоставных вопросах играет роль логической связки **и**. Примеры сложносоставных вопросов:

**? мама (х, у), мама** **(у, Оля)** - *Кто мама у мамы Оли?*

**х** = **Зина у = Люба**

**? мама (х,** у), **папа** **(у, Оля)** - *Кто мама у папы Оли?*

**НЕТ**

**Правила** в Прологе - это правила логического вывода. Слева в правилах записывается следствие, а справа - предусловие. Пред­условие может состоять из одного или нескольких предикатов, раз­деляемых запятыми. Примеры записи правил вывода на Прологе:

**студент (х) ← занятие** **(х, учеба);** - *Студент - тот, кто* *занят учебой*;

**нумизмат (х) ←собирает (х, монеты);** - *Нумизмат - тот, кто* *собирает монеты.*

Примеры вопросов на использование этих правил:

**? студент (х)** - *Кто - студент?*

**х** = **Алеша**

**х = Лена**

**? нумизмат** **(у)** - *Кто* - *нумизмат?*

**у** = **Алеша**

В правилах со сложносоставными определениями запятая также играет роль логической связки **и**, объединяя условия, образующие определение. Такого рода правила позволяют создавать самые слож­ные и изощренные базы знаний по самым различным предметным областям и применениям.

Приведем пример составления базы знаний о друзьях. Будем раз­личать друзей по их именам: Алеша, Оля и т. д. Включим в базу дан­ных следующие сведения о друзьях:

что им нравится;

что они коллекционируют;

чем они занимаются;

какие оценки они имеют.

При такой постановке проблемы и предметная область, и круг основных вопросов очерчены достаточно четко. Для записи фактов на Прологе примем следующие предикаты:

**нравится (<имя>, <вещь>);**

**собирает (<имя>, <вещь>);**

**занимается (<имя>, <предмет>);**

**оценка (<имя>, <предмет>, <балл>);**

Вместо **<имя>, <вещь>, <предмет>, <балл>** при составлении базы знаний необходимо подставить конкретную информацию о конкрет­ных друзьях. Записывать имена будем с большой буквы в имени­тельном падеже. Далее, **<вещь>** и **<предмет>** - это существительные в именительном падеже, **<балл>** - целое число от 1 до 5.

Пусть об Оле и Алеше известно следующее:

1. Оле нравится музыка. Она собирает фотографии любимых пев­цов. Занимается домоводством. Оля имеет 4 по русскому языку и 5 по алгебре.

2. Алеше нравится история, он собирает монеты, естественно, име­ет 5 по истории, занимается в археологическом кружке.

Соответствующая база данных на языке Пролог:

**нравится (Оля, музыка);** - *Оле нравится музыка*

**нравится (Алеша, история);** - *Алеше нравится история*

**собирает (Оля, фотографии);** - *Оля собирает фотографии*

**собирает (Алеша, монеты);** - *Алеша собирает монеты*

**собирает (Алеша, значки);** - *Алеша собирает значки*

**оценка (Оля, русский, 4);** - *Оля имеет 4 по русскому языку*

**занимается (Алеша, бизнес);** - *Алеша занимается бизнесом*

**оценка (Оля, алгебра, 5); -** *Оля имеет оценку 5 по алгебре*

**оценка (Алеша, история,** 5); - *Алеша имеет оценку 5 по истории*

К составленной базе данных можно обращаться с самыми разными вопросами об интересах, занятиях, склонностях и успехах в учебе. Примеры самых простых вопросов и ответов, получаемых от ЭВМ:

**? занимается (Алеша, футбол)** - *Занимается ли Алеша футболом?*

**нет**

**? нравится (Оля, музыка)** - *Нравится ли Оле музыка?*

**да**

Использование в вопросах переменных позволяет получать от ЭВМ информацию, хранящуюся в базе данных. В ответ на такие вопросы выводятся все возможные варианты ответов. Например:

**? нравится** (**х**, **у)** - *Кому что нравится?*

**х** *=* **Оля у = музыка**

**х *=* Алеша у *=* история**

Если какая-то часть информации по той или иной причине не нужна, то вместо соответствующей переменной в вопросе ставится знак подчеркивания «\_»:

**? собирает (\_ , х)** - *Что собирают друзья?*

**х** = **фотографии**

**х *=* монеты**

**х = значки**

Наконец, в вопросах можно одновременно использовать как пе­ременные, так и конкретные значения. Например:

**? занимается (х, музыка)** - *Кто занимается музыкой ?*

**нет**

**? занимается (Алеша, у) -** *Чем занимается Алеша ?*

**у = бизнес**

**? собирает (х, монеты) -** *Кто собирает монеты ?*

**х = Алеша**

**? оценка (х,** **\_ , 5)** *- Кто имеет пятерки?*

**х = Оля**

**х = Алеша**

Примеры сложносоставных вопросов:

1. Кто занимается бизнесом и собирает монеты?

**? занимается (х, бизнес), собирает (х, монеты)**

**х = Алеша**

2. Какие оценки имеет тот, кто собирает монеты?

**? собирает (х, монеты), оценка (х, р, z)**

**х = Алеша**

**р = история**

**z = 5**

К составленной базе данных можно добавить следующие правила вывода:

**книголюб (х) ← нравится (х, книги),** - *Книголюб - тот, кто*

**собирает (х, книги)** *любит и собирает книги*

**бизнесмен (х) ← собирает (х, монеты),** - *Бизнесмен - тот, кто*

**занятие (х, бизнес)** *собирает монеты и занима­* *ется бизнесом*

Примеры использования правил-определений:

**? книголюб (х)** - *Кто - книголюб?*

**НЕТ**

**? бизнесмен** (**у**) - *Кто - бизнесмен ?*

**у = Алеша**

**В о п р о с ы**

1. Как записываются факты на языке Пролог?

2. Как записываются вопросы на языке Пролог?

3. Как записываются правила в языке Пролог?

**З а д а ч и**

1. Опишите на языке Пролог данные о своей семье: о маме, папе, сестрах, братьях, дедушках и бабушках - кто кому приходится мамой и папой. Укажите вопросы для определения родственных отношений.

2. Опишите данные о своих друзьях с указанием их увлечений (кто что любит), занятий (кто чем занимается). Подберите правила для оп­ределения понятий:

а) сластена; д) спортсмен;

б) филателист; е) бизнесмен;

в) математик; ж) музыкант;

г) программист; з) мусорщик.

3. Опишите в форме фактов для себя и своих ближайших друзей-одноклассников данные об оценках по предметам:

а) литература; г) физкультура;

б) математика; д) информатика;

в) физика; е) история.

4. Подберите правила определения понятий:

а) математик; д) физик;

б) историк; е) лирик;

в) двоечник; ж) троечник.

г) отличник;

## 3.4. Базы знаний на ЭВМ

**Экспертные системы** и базы знаний на ЭВМ - одно из перспек­тивных направлений в области искусственного интеллекта. Такие экспертные системы в ближайшем будущем станут интеллектуаль­ными ассистентами людей во многих областях профессиональной деятельности.

**Базы знаний** - это совокупность фактов и правил вывода, храня­щихся в памяти ЭВМ. Содержание любой базы знаний составляют конкретные и обобщенные факты и сведения об определенной предметной области или сфере деятельности. Примеры предметных областей - семья, школа, вуз, магазин, рынок, ферма, фирма, завод, офис, банк и т.п.

**Конкретные сведения** в базах знаний представляются фактами, которые записываются в виде предикатов с конкретными значениями. Примеры записи фактов в форме предикатов:

**мама (Люба, Оля);** - *Люба* - *мама Оли*

**оценка (Вова, физика,** **5);** - *Вова имеет 5 по физике*

**Обобщенные сведения** в базах знаний записываются в форме правил вывода, выражающих определения понятий. Примеры обоб­щенных сведений:

**бабушка (х, z) ← мама (х,** **у), мама (у, z)** - *бабушка - это мама* *мамы*

**двоечник** **(х) ← оценка (х,** \_ ,**2)** - *двоечник - тот,* *у кого есть двойки*

Используя совокупность конкретных и обобщенных фактов и правил, компьютеры могут **давать ответы** на широкий круг осмыс­ленных вопросов. Для этого соответствующая база знаний с соот­ветствующей системой понятий должна храниться в памяти ЭВМ.

**Базы знаний** могут содержать правила вывода следующих видов:

- правила определения понятий;

- правила принятия решений;

- способы решения задач;

- правила поведения и т. п.

При определении новых понятий необходимо различать их объем и содержание. **Объем понятия** - это совокупность вещей или явлений, отвечающих данному понятию. Объем понятий соотносит словам и терминам соответствующие объекты предметной области.

**Содержание понятия** - это совокупность логических признаков, выделяющих объекты, отвечающие данному понятию, среди других объектов. С логической точки зрения определение понятия пред­ставляет описание совокупности признаков, характеризующих его содержание.

Приведем примеры определения понятий.

1. Понятие «мама». Объем понятия - совокупность всех мам. Содержание понятия - женщина, имеющая детей. Формализация понятия на Прологе может выражаться конкретными фактами. При­меры:

**мама (Люба, Оля);** - *Люба - мама Оли*

**мама (Зина, Люба);** - *Зина - мама Любы*

2. Понятие - «бабушка». Объем понятия - совокупность всех бабушек. Содержание понятия - *«бабушка - это мама мамы или папы».* Формализация этого понятия на Прологе:

**бабушка** **(х, z) ← мама (х,** **у), мама (у, z);** - *бабушка - это мама* *мамы*

**бабушка (х, z) ← мама (х, у), папа** **(у, z);** - *бабушка - это мама* *папы*

3. Понятие «музыкант». Объем понятия - совокупность людей, занимающихся музыкой. Содержание понятия - «музыкант - чело­век, который любит музыку и занимается музыкой». Это понятие на языке Пролог можно записать в виде правила:

**музыкант (х) ← любит (х, музыка), занятие (х, музыка).**

4. Понятие «студент». Объем и содержание понятия - учащиеся в вузах, университетах и колледжах. Определение этого понятия на Прологе можно записать в виде набора правил:

**студент (х) ← занятие (х, учеба), место (х, университет);**

**студент (х) ← занятие (х, учеба), место (х, институт);**

**студент (х) ← занятие (х, учеба), место (х, колледж);**

Для создания **сложных баз знаний** может понадобиться целая сис­тема понятий, характеризующих объекты в заданной предметной области. Эта система определений базируется на наборе признаков, где каждый признак является характеристикой некоторого объекта.

Примеры признаков объектов и соответствующих значений:

цвет - красный, белый, зеленый, черный и т.д.;

вес - определяется в килограммах;

возраст - определяется в годах: 1, 2, 3, ...

Примеры записи признаков на Прологе:

**возраст (Иванов, 18);**

**вес (Иванов, 85);**

**цвет (Иванов, глаза, синий);**

**цвет (Иванов, волосы, белый);**

**Основные возможности** баз знаний:

- поиск ответов на сложные вопросы;

- логическая обработка данных;

- моделирование процедур принятия решений;

- обновление и ввод дополнительных данных;

- вывод информации в естественно-языковой форме;

- создание новых баз знаний.

**В о п р о с ы**

1. Что такое базы знаний?

2. Как записываются факты на языке Пролог?

3. Как записываются вопросы на языке Пролог?

4. Как записываются правила на языке Пролог?

5. Что такое содержание понятий?

6. Каковы основные возможности баз знаний?

**З а д а ч и**

1. Составьте базу знаний о книгах в библиотеке. Включите следующие сведения: название книги, жанр книги, автор, главный герой, страна.

2. Составьте базу данных о собаках и кошках у ваших знакомых. В базу включите сведения об их породе, возрасте, кличках.

3. Составьте базу знаний о соревнованиях. База должна содержать две группы фактов.

Первая группа фактов - сведения о командах: названия команды, города, тренер.

Вторая группа фактов - сведения о матчах: даты, команды и счет.

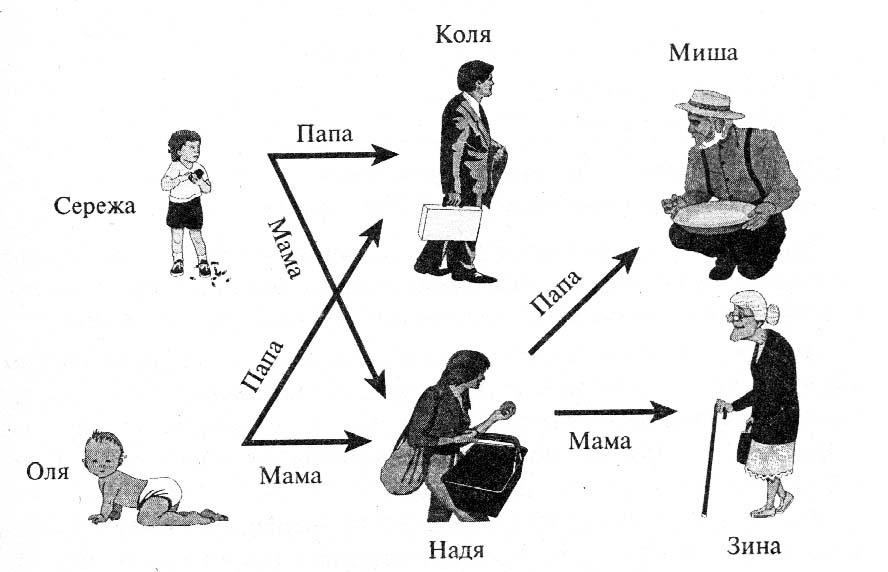
## 3.5. Законы логического вывода

**Умение делать выводы** - основная способность любого интеллек­та. Основным признаком интеллекта являются знания и умения ре­шать определенные задачи. Основой этих интеллектуальных умений являются законы и принципы логического вывода.

Законы человеческого мышления изучаются логикой и психоло­гией. Принципы **искусственного интеллекта** разрабатываются в мате­матической логике и информатике на основе методов машинного доказательства теорем и постановки экспериментов на ЭВМ.

**Знание законов** логического вывода позволяет выверять правиль­ность рассуждений, доказательств, а также находить ошибочные утверждения. Незнание или нарушение этих законов и принципов приводит к логическим ошибкам и принятию неправильных решений.

В качестве иллюстраций рассмотрим базу знаний о семье. Пусть в семье есть дочь Оля, сын Сережа, мама Надя, папа Коля, бабушка Зина и дедушка Миша:



Соответствующая база данных на языке Пролог:

**мама (Зина, Надя); папа (Миша, Надя);**

**мама (Надя, Оля); папа (Коля, Оля);**

**мама (Надя, Сережа); папа (Коля, Сережа);**

Система Пролог при наличии этих фактов в памяти ЭВМ даст следующие ответы на вопросы о «папах» и «мамах»:

**? мама (Надя, Оля)** - *Надя* - *мама Оли?*

**да**

**? папа (Сережа, Надя)** - *Сережа - папа Нади?*

**нет**

Машина, использующая систему Пролог, дает ответы строго в соответствии с определенными логическими законами и принципа­ми логического вывода. Разберем эти законы и принципы.

**Законы логики** - это общие законы логического вывода, исполь­зуемые для вывода различных умозаключений. Наличие интеллекта и у компьютеров и у людей определяется прежде всего способностя­ми делать выводы и умозаключения в различных ситуациях.

**Закон противоречия** - первый общий закон логики. Этот закон впервые высказал Аристотель - основатель логики как научной дисциплины:

А **и не** А **- Не** могут быть **истинны** одновременно

**ложь** суждение и его отрицание

Примеры противоречивых утверждений:

1) *«яблоко - спелое»;*

*«яблоко - неспелое»;*

2) *«треугольник АВС - прямоугольный»;*

*«стороны треугольника равны друг другу».*

Высказывание противоречивых суждений и утверждений являет­ся примером серьезных **логических ошибок.** Наличие противоречий говорит о некотором умысле либо о нарушениях в сознании.

В системе Пролог закон противоречия лежит в основе **механизмов вывода** ответов на вопросы. А именно, система Пролог для каждого вопроса формирует его отрицание и для этого отрицания ищет про­тиворечие по отношению к фактам и правилам в базе знаний, хра­нящейся в памяти.

Так, вопрос **? мама (Надя, Оля)** ЭВМ преобразует в отрицание **не мама (Надя, Оля)** и временно присоединяет его к базе знаний. При этом получившаяся база становится противоречивой и система Про­лог объявляет отрицание ложным и выводит ответ ДА.

**Закон исключения третьего** - второй общий закон логики, ука­занный Аристотелем:

А **или не** А **- Истинно** суждение либо его отрицание,

**истина третьего не дано.**

Примеры взаимоисключающих утверждений:

1) «Сегодня будет дождь» или «Сегодня дождя не будет».

2) «Любой треугольник правильный либо в нем есть разные сто­роны».

**Принципы логического вывода** в системе Пролог отличаются от законов Аристотеля, поскольку его законы выражаются на языке высказываний, а работа системы Пролог основана на использова­нии предикатов.

Соответствующая запись законов логики на языке исчисления пре­дикатов представляет конструктивные процедуры, позволяющие осу­ществлять конкретные логические выводы из общих правил и кон­кретных фактов.

**Закон противоречия** в записи на языке исчисления предикатов преобразуется в процедуру вывода ответов на простейшие вопросы вида **? А(с):**

А(х) **и не** А(с) - При **противоречии** А(х) **и не** А(с)

**ложь,** при х = с контрпримером служит х = с.

Содержательный смысл: При противоречии **А(х)** и **не А(с)** контр­примером служит **х = с**. Здесь **х** - переменная, а **с** - конкретное значение, при котором отрицание ответа оказывается ложным. Это значение **х = с** выводится системой Пролог в качестве ответа на поставленный вопрос. Примеры такого рода вопросов и ответов:

**? папа (х, Коля)** - Кто папа у Коли ?

**нет**

**? папа (х, Оля)** - Кто папа у Оли ?

**х = Коля**

**? мама (х, у)** - Что известно о мамах?

**х = Зина у = Надя**

**х = Надя у = Оля**

**Закон двойного отрицания** - третий общий закон логики. Этот закон также был высказан Аристотелем:

**не** (**не** А) - Если **неверно отрицание,**

А то суждение – истинно

Примеры рассуждений.

1) Неверно, что «*Вчера не было дождя*».

Следовательно, «*Вчера был дождь*».

2) Неправда, что «*это сделал не Саша*».

Следовательно, «*это сделал Саша*». (?)

Из второго примера видно, что закон двойного отрицания явля­ется косвенным доказательством, поскольку оно не опирается на факты или аргументы. По этим причинам закон двойного отрицания может оказаться ошибочным и этот закон не является **общезначи­мым** (верным для всех случаев и ситуаций).

Правильность утверждений и рассуждений даже при безупречной логике доказательств зависит от достоверности исходных фактов и положений. Эту идею выражает четвертый общий логический за­кон - закон достаточных оснований, впервые сформулированный Лейбницем - создателем одной из самых первых механических вы­числительных машин и основателем исчисления предикатов.

**Закон достаточных оснований:**

**Всякое утверждение** должно предполагать существование аргументов и фактов, достаточных для его обоснования.

Иными словами, любое утверждение должно предполагать нали­чие набора конкретных фактов и правил, из которых должно выте­кать утверждаемое. **Нарушениями** это закона являются рассуждения, опирающиеся на недостоверные факты или положения, истинность которых не проверяется, а принимается на веру.

Пример рассуждений, не имеющих достаточных оснований:

1. «*Если дорогу перебежала черная кошка, то быть неприятностям*».

2. «*Это верно, потому что это - справедливо*».

**Экспертные системы** на ЭВМ с этой точки зрения должны созда­ваться исключительно на основе достоверных данных и общих пра­вил вывода, проверенных практикой. Включение в базу знаний не­достоверных данных или неподтвержденных обобщений и правил может привести к появлению ошибок и получению неправильных решений.

Причина такой работы экспертных систем состоит в том, что компьютеры могут делать **выводы** и **умозаключения** только исходя из фактов и правил, имеющихся в базе знаний, и только из этих дан­ных - и ничего другого. Практическую ценность для систем ма­шинного интеллекта представляют принципы логического вывода на основе не только фактов, но и правил.

**Закон логического вывода** modus ponens - первый закон, указан­ный Аристотелем:

**А → В, А -** Если из А **следует** В и посылка А **истинна,**

**В** то выполняется и следствие В.

Примеры применения правила логического вывода.

1. «*Если идет дождь, то на улице мокро*».

«*Пошел дождь*».

Следовательно, «*на улице мокро*».

2. «*Если кому-то весело, то он улыбается*».

«*Маша развеселилась*».

Следовательно, «*Маша улыбается*».

В предикатной форме закон логического вывода превращается в процедуру вывода новых сведений из имеющихся общих правил и конкретных фактов:

**А(х)** **→ В(х),** **А(с)** - Если из А(х) **следует** В(х) и А(х) выполнено

**В(с)** для х = с, то выполняется В (с).

Данная процедура из конкретного факта **А(с)** и общего правила логического вывода **А(х) → В(х)** позволяет **вывести новое** конкрет­ное утверждение **В(с).** Иными словами, эта процедура вывода позво­ляет выводить новые конкретные сведения из общих правил и уже известных конкретных фактов.

Следовательно, знание и применение законов логического выво­да позволяет порождать **новые сведения,** являющиеся логическим следствием уже известных сведений. Этим свойством и этими зако­нами могут пользоваться не только люди - они применяются в си­стемах машинного интеллекта.

**Закон вывода отрицаний** - второй закон логического вывода:

**А → В, не В** - Если из А **следует** В, но следствие В ложно,

**не** **А** то **не выполняется** посылка А.

Примеры логического вывода из отрицаний.

1. «*Если идет дождь, то на улице мокро*».

«*На улице сухо*».

Следовательно, «*На улице не было дождя*».

2. «*Если кому-то весело, то он улыбается*».

«*Вова не улыбается*».

Следовательно, «*Вове не весело*».

В предикатной форме закон вывода отрицаний превращается в конструктивную процедуру вывода отрицаний:

**А(х) → В(х), не В(с)** - Если из А(х) **следует** В(х), но В **не выполнено**

**не А(с)** для х = с,то **не выполняется** и А(с).

Данная процедура из конкретного отрицания следствия не **В(с)** и общего правила **А(х) → В(х)** выводит конкретное отрицание **не А(с)** предпосылки при **х = с**. Эта процедура используется в системе Пролог как основной механизм поиска ответов на сложные вопросы, в ко­торых участвуют определяемые предикаты.

В качестве иллюстраций дополним рассмотренную базу знаний о семье набором правил на языке Пролог, выражающих понятие «ро­дитель»:

**родитель (х, у) ← мама** **(х, у);** - *Мама - родитель*

**родитель (х, у) ← папа (х, у);** - *Папа – родитель*

После ввода этих правил в ЭВМ система Пролог на вопросы о родителях выдаст такие ответы при указанной выше базе данных:

**? родитель (Надя, х)** - *Кому родитель Надя* ?

**х** = **Оля**

**х = Сережа**

Вывод ответов на эти вопросы система Пролог проводит следу­ющим образом. Во-первых, вопрос **? родитель (Надя, х)** будет заменен на отрицание **не родитель (Надя,** **х).** Далее это отрицание будет сопо­ставлено с правилом вывода **родитель** **(х, у) ← мама** **(х, у),** а затем с правилом родитель **(х, у) ← папа** **(х, у)**.

Применение этой же процедуры вывода ко второму определению **родитель (х, у) ← папа** **(х, у)** даст отрицание **не папа (Надя,** **х)**, озна­чающее утверждение «Надя не является папой никому». Для этого утверждения компьютер не имеет в базе данных никаких соответст­вующих фактов и после просмотра базы знаний по этому варианту выдает ответ НЕТ.

Применение к отрицанию **не родитель (Надя, х)** и определению **родитель (х, у) ← мама** **(х, у)** рассматриваемой процедуры приводит к выводу утверждения **не мама (Надя, х),** означающему «Надя не является мамой никому». Для этого отрицания машина найдет два конкретных противоречащих ему факта **мама (Надя, Оля)** и **мама (Надя, Сережа).** Используя конструктивную процедуру вывода отве­тов из отрицаний, компьютер даст два конкретных ответа **- х = Оля** и **х = Сережа.**

**Закон тождества** - четвертый общий логический закон, указанный Аристотелем:

**«Предмет** рассмотрения должен быть определен

и не должен меняться до конца обсуждения»

Данный закон носит **фундаментальный характер** для работы экспертных систем - правильные выводы и решения могут быть получены от экспертных систем только при строгом совпадении определений вещей из рассматриваемой предметной области.

**Расхождения в понимании** и определении предметных понятий могут приводить и, как правило, приводят к логическим ошибкам и получению неправильных выводов и результатов, что наблюдается среди людей, не обладающих необходимыми профессиональными знаниями.

Примером нарушения закона тождества является **подмена пред­мета,** когда два собеседника осознанно или неосознанно говорят о разных вещах, что приводит их к непониманию, спорам и разногла­сиям. Классический пример нарушения - ситуация: «*я* - *про Фому, а он - про Ерему».*

В **системе Пролог** и в системах искусственного интеллекта вывод ответов на сложные вопросы основан на принципе унификации (взаимосогласования) ответов. По этой причине экспертные системы на ЭВМ в отличии от людей могут производить вывод ответов на сложные вопросы только в соответствии с принципом унификации.

**Принцип унификации** ответов состоит в том, что общие перемен­ные во взаимосвязанных вопросах должны получать одинаковые значения. Пример ответа на сложносоставной вопрос, состоящий из двух подвопросов:

**? мама (z, у), мама** **(у**, **Оля)** - *Кто мама у мамы Оли?*

**z** *=* **Зина**

**у = Надя**

Вывод ответов на сложносоставные вопросы состоит в выделе­нии подвопросов и поиске на них ответов по частям:

**? мама (z, у), мама (у, Оля)**

⁄ \

**? мама (z,** **у) ?мама** **(у**, **Оля)**

⁄ \ ⁄

**z = Зина у = Надя**

В данном примере **общим элементом** в выделяемых подвопросах ? **мама** (z, у) и ? **мама** (у, Оля) является переменная «у». Ответом на первый подвопрос ? **мама** (z, у) будут значения z = Зина и у = Надя. Ответы на второй подвопрос ? **мама** (у, Оля) в соответствии с прин­ципом взаимосогласования будет проводиться для значения у = Надя.

Принцип вывода взаимосогласованных ответов в системе Пролог распространяется и на сложносоставные правила, включаемые в базы знаний и процедуры логического вывода. Приведем примеры слож­ных определений:

**бабушка (z, х) ← мама** **(z, у), мама** **(у, х)**;

**бабушка (z, х) ← мама (z,** **у), папа** **(у, х)**;

**дедушка (z, х) ← папа (z, у), мама (у, х);**

**дедушка (z, х) ← папа (z, у), папа (у, х);**

При наличии этих правил в памяти ЭВМ можно получить следу­ющие вопросы о бабушках и дедушках:

**? бабушка (z, Оля)** - *Кто бабушка у Оли?*

**z** *=* **Зина**

**? дедушка (z, Надя)** - *Кто дедушка у Нади?*

**нет**

Принцип унификации в системе и языке Пролог является **общим механизмом** логического вывода ответов на сложные вопросы в базах знаний на Прологе и тем самым - конструктивной реализацией закона тождества для машинных систем искусственного интеллекта. Таким образом работа экспертных систем на ЭВМ основана строго на выполнении **требований законов логики** - закона тождества и за­кона достаточных оснований.

**В о п р о с ы**

1. В чем состоит закон противоречия?

2. В чем заключается закон исключения третьего?

3. Как используется закон вывода следствий?

4. Как используется закон отрицания следствий?

5. В чем недостатки закона двойного отрицания?

6. В чем состоит закон достаточных оснований?

7. В чем заключается закон тождества?

8. В чем состоит принцип унификации?

9. Как согласуются ответы на сложные вопросы?

**3 а д а н и я**

1. Укажите примеры двойного отрицания для утверждений:

а) «сегодня был дождь»; в) «х = 2» **и** «х = 3»;

б) «х = 0» **или** «у = 0»; г) «5 не делится на 2 и на 3».

2*.* Пусть утверждение А «прошел дождь», а утверждение В «на улице сыро». Истинны ли следующие суждения?

а) А ⇒ В (прямое доказательство);

б) В ⇒ А (обратное доказательство);

в) **не** А ⇒ **не** В (противоположное доказательство);

г) **не** В⇒ **не** А (противоположное обратному).

3. Предложите систему признаков и понятий для описаний

а) класса «Млекопитающие»; в) класса «Рыбы»;

б) класса «Птицы»; г) класса «Насекомые».

В систему понятий введите следующие признаки: окрас, продолжи­тельность жизни, умения ползать, летать, ходить и т. д.

4. Составьте базу знаний по всемирной географии. В базу знаний включите сведения о странах: название столицы, число жителей, тип государства, размеры страны, континент.

5. Составьте базу знаний по городам своей страны. В базу знаний включите сведения о размерах городов, числе жителей, расстоянии от столицы, названии самых крупных заводов, фабрик, музеев, стадионов и т. п.

6. Составьте базы данных

а) по литературе; г) по истории;

б) по зоологии; д) по ботанике;

в) об автомобилях; е) по кулинарии.

# Глава 4. ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

## 4.1. Основные свойства алгоритмов

Алгоритм относится к **фундаментальным понятиям** информатики. На понятии алгоритма построено все основные принципы програм­мирования - составления программ для вычислительных машин.

**Алгоритм** - это совокупность действий со строго определенными правилами выполнения. В информатике изучаются различного рода алгоритмы - диалоговые алгоритмы, алгоритмы обработки данных, вычислительные алгоритмы, алгоритмы управления роботами, стан­ками и другими техническими устройствами.

Пример диалогового алгоритма:

**Алгоритм Блок-схема**

***алг «приветствие»* ↓**

***нач запрос («Ваше имя=», NN)***

***запрос («Ваше имя=», NN)*  ↓**

***вывод («Добрый день», NN) вывод («Добрый день», NN*)**

***кон* ↓**

Для **описания алгоритмов** используются блок-схемы, изображен­ные справа, или структурированная запись, приведенная слева. Блок-схемы наглядны. Однако блок-схемы трудно рисовать, в них сложно вносить изменения и исправления из-за сложности перерисовки рамок и стрелок. Однако блок-схемы до сих пор требуются отечест­венными стандартами на документирование программ.

Достоинство записи алгоритмов и программ в структурирован­ной форме заключается в **простоте их чтения и ввода** с экрана ЭВМ, а также в простоте внесения изменений и исправлений с исполь­зованием даже самых простейших редакторов тестов. По этим при­чинам зарубежом блок-схемы уже давно не используются ни для документирования, ни для обучения, а все современные языки построены на принципах структурного программирования.

Приведем примеры описания алгоритма и программы в структу­рированной записи:

**Алгоритм Программа**

***алг «приветствие»* ' приветствие**

***нач* сls**

***запрос («Ваше имя=», NN)* input «Ваше имя=», NN$**

***вывод («Добрый день», NN)* print «Добрый день», NN$**

***кон* end**

Алгоритм, приведенный слева, записан на псевдокоде. **Псевдокод** - это язык записи структурированных алгоритмов в качестве докумен­тации к программам для ЭВМ. Особенность псевдокода заключается в том, что описания на нем выполняются на родном языке — рус­ском, английском, украинском, казахском, немецком и т. п.

Программа, приведенная справа, записана на языке **Бейсик** - языке программирования персональных ЭВМ. Языками программи­рования называются формализованные языки, используемые для записи программ на ЭВМ. Одним из них является язык Бейсик.

**Достоинства псевдокода** заключаются в том, что описания алго­ритмов, записанные на родном языке, намного проще читать и по­нимать, чем запись программ на языке с иностранной лексикой. По этим причинам псевдокод используется как основное средство до­кументирования программ во всех ведущих фирмах, занимающихся разработкой программ.

С точки зрения информатики алгоритмы, записанные в такой обобщенной записи, позволяют выразить общую **логику работы про­грамм,** независимо от используемых языков программирования и типов ЭВМ. При этом алгоритмы, записанные в такой обобщенной форме, могут быть реализованы с помощью различных языков про­граммирования для самых различных типов ЭВМ.

В качестве примера приведем реализацию этого же диалогового алгоритма на самой ранней версии языка Бейсик, использовавшего­ся на самых первых персональных компьютерах:

**Алгоритм Программа**

***алг «приветствие»* 10 ' приветствие**

***нач* 20 сls**

***запрос («Ваше имя=», NN)* 30 input «Ваше имя=», NN$**

***вывод («Добрый день», NN)* 40 print «Добрый день», NNS**

***кон* 50 end**

**Основные свойства** алгоритмов и программ для вычислительных машин - однозначность, результативность, правильность и массо­вость. Этими свойствами алгоритмы отличаются от различного рода расплывчатых и неоднозначных предписаний, инструкций и кули­нарных рецептов, которые могут толковаться и исполняться многими способами.

**Однозначность** алгоритмов - это однозначность правил их вы­полнения. Следствием этого свойства алгоритмов является однознач­ность результатов их выполнения в одинаковых начальных условиях. Это не всегда верно для кулинарных рецептов, когда разные испол­нители в одних и тех же условиях могут придавать различный вкус и пикантность одним и тем же блюдам.

**Результативность** - это завершение выполнения алгоритмов оп­ределенными результатами. Результативность - наиболее важное свойство алгоритмов и программ, предназначенных для решения при­кладных задач. Алгоритмы и программы, не дающие результатов или ведущие к сбоям и отказам, никому не нужны.

**Массовость** - это возможность применения алгоритмов в раз­личных конкретных исходных условиях. Массовые алгоритмы осо­бенно важны для решения прикладных задач, когда алгоритмы и программы должны обеспечить решение целого класса задач, разли­чающихся исходными данными.

**Правильность алгоритмов** определяется правильностью резуль­татов, получаемых с их помощью. По этой причине правильность алгоритмов и программ является относительным понятием. Оценка правильности может проводиться только при наличии требований к конечным результатам.

**Алгоритм** считается **правильным,** если он дает правильные резуль­таты для любых допустимых начальных условиях. Правильность алгоритмов гарантирует правильность результатов их выполнения.

**Алгоритм** содержит **ошибки,** если его выполнение может привести к отказам, сбоям или неправильным результатам, либо вовсе не дает никаких результатов. Эти ошибки называются алгоритмическими. Алгоритмы и программы, содержащие такие ошибки, могут нанести вред или ущерб тем, кто захочет ими воспользоваться.

Для оценки правильности алгоритмов и программ необходимо уметь оценивать результаты выполнения составляющих их действий и конечные результаты их выполнения в целом.

Простейшие виды машинных операций - операции присваивания. С помощью присваивании в алгоритмах описываются вычисления в программах для ЭВМ. Рассмотрим примеры операций присваива­ния и описания результатов их выполнения.

**Присваивания: Результаты:**

**а := 0 а = 0**

**b := а + 1 b ' = а + 1 = 1**

**b := b + 1 b " = b' + 1 = 2**

Запись присваиваний читается:

**а := 0** - «переменной **а** присвоить значение **0»;**

**b := b + 1** - «переменной **b** присвоить значение **b + 1**».

Записи в колонке результатов читаются так:

**а = 0** - «значение **а** равно **0**»;

**b'** = **b** + **1** - «значение **b'** равно **b + 1**».

Здесь **а** и **b - программные переменные** - область машинной па­мяти, в которой хранятся их значения **а** и **b**. В отличии от обычных математических переменных программные переменные могут полу­чать новые значения. В частности, присваивание **b: = b + 1** запи­сывает в программную переменную **b** новое значение **b',** равное величине **b + 1**, где **b** - прежнее значение переменной **b**.

Для описания результатов выполнения алгоритмов и программ могут и должны использоваться **спецификации.** Спецификации - это точные, математически строгие описания. Примерами спецификаций могут служить сценарии диалоговых программ.

**Сценарии** диалоговых алгоритмов и программ - это совокупность текстов, картинок и сообщений, появляющихся на экранах ЭВМ. Рассмотрим в качестве примера сценарий алгоритма рисования домика на экране ЭВМ.

Сценарий **«Домик»**

Решение - следующие алгоритм и программа, результатом работы которых должен быть приведенный выше рисунок:

**Алгоритм Программа**

***алг «Домик» '* Домик**

***нач* screen 2,0**

***линия (130,40)-( 100,100), красная* line (150,40)-(100,100),8**

***линия (130,40)-(200,100), красная* line (150,40)-(200,100),8**

***рамка(100,100)-(200,200), белая* line (100,100)-(200,200),15,b**

***рамка(130,120)-(170,160), синяя* line (130,120)-(170,160),3,b**

***кон* end**

Однако результатом выполнения приведенных алгоритма и про­граммы будет следующий рисунок:

Экран ЭВМ

Причиной того, что на этом рисунке крыша «поехала» влево, являются алгоритмические ошибки - неправильный расчет коорди­нат крыши в алгоритме, из-за чего составленная программа дает не тот рисунок, который указан в сценарии.

Примером прикладного алгоритма и программы может служить следующий алгоритм расчета прибыли:

**Алгоритм Программа**

***алг «расчет прибыли»* ' расчет прибыли**

***нач* сls**

***запрос («доходы =», d)* input «доходы =», d**

***запрос («расходы =», r)* input «расходы =», r**

***р:* = *d - r* р = d - r**

***вывод («прибыль =», р)* print «прибыль *=»,* р**

***кон* end**

Сценарий диалога Протокол диалога

***доходы =?<d>* доходы =? 1000**

***расходы* =? <г> расходы =? 700**

***прибыль = <р>* прибыль = 300**

Для проверки правильности алгоритма и программы необходима постановка задачи. Приведем строгую постановку решаемой задачи.

Задача: расчет прибыли.

Треб.: **р** - прибыль.

Дано: **r** - расходы;

**d** - доходы.

Где: **d = r + р**.

При: **d > 0.**

Для оценки правильности полученных результатов нужно сверить расходы и прибыль с доходами. В нашем случае это должно быть 700 + 300 = 1000, что выражает правильный конечный результат при указанных данных.

Для оценки правильности алгоритма и программы необходимо рассмотреть конечные результаты их выполнения при произвольных значениях данных d и г. Вычисляемая величина р по алгоритму будет равна

Операция Результат

**р := d - r р = d – r**

Подставляя в условие постановки задачи это значение, получаем:

**d = r + p = r + (d - r) = d** - верное тождество.

Таким образом, при любых значениях исходных данных результаты выполнения приведенного алгоритма будут правильными.

**В о п р о с ы**

1. Что такое алгоритм?

2. Каковы основные виды алгоритмов?

3. Что такое однозначность алгоритмов?

4. Что такое результативность алгоритмов?

5. Что такое правильность алгоритмов?

6. Что такое массовость алгоритмов?

7. Что такое алгоритмические ошибки?

**З а д а ч и**

1. Составьте сценарий, алгоритм и программу:

а) поздравления с Новым годом;

б) поздравления с Днем рождения;

в) регистрации даты рождения;

г) регистрации фамилии и имени.

2. Составьте сценарии диалога, алгоритм и программу:

а) расчета сдачи за товар;

б) расчета остатка от прибыли;

в) пересчета рубль/доллар;

г) расчета остатка времени до 18.00.

3. Составьте сценарий, алгоритм и программу вычислений:

а) времени движения по длине пути и скорости;

б) длины пути по времени и скорости движения;

в) средней скорости по времени и длине пути.

4. Составьте картинки, алгоритмы и программу рисования:

а) российского флага; г) украинского флага;

б) шведского флага; д) французского флага;

в) японского флага; е) британского флага.

5. Составьте сценарий, алгоритмы и программу на Бейсике вывода изображений:

а) яхты; д) автомобиля;

б) трактора; е) усадьбы;

в) дерева; ж) цветка;

г) рыбы; з) птицы.

## 4.2. Базовые средства программирования

Базовыми средствами программирования для персональных компьютеров считаются языки семейства **Basic (Бейсик).** Эти языки программирования имеются на всех персональных компьютерах и широко используются для обучения началам программирования в школах и вузах.

Бейсик является примером одного из лучших языков диалогового программирования для ЭВМ. По этой причине Бейсик оказался самым первым языком программирования самых первых персональ­ных компьютеров, созданных фирмой Microsoft.

На персональных компьютерах IBM PC язык Бейсик имеет три версии, связанные с операционными системами для этих компьюте­ров, созданных и развиваемых фирмой Microsoft:

1) традиционный Бейсик (без ОС),

2) структурный Бейсик(МS DOS),

3) графический Бейсик (Windows).

**Традиционный Бейсик** полностью воспроизводит язык програм­мирования самых первых персональных компьютеров, на которых отсутствовали операционные системы. В связи с прекращением про­изводства этих компьютеров данная версия языка Бейсик потеряла свое прежнее значение и не используется на современных ЭВМ.

**Структурный Бейсик** под именем **Quick Basic** был создан вместе с первыми моделями персональных компьютеров IBM PC как базовое средство программирования в операционной системе MS DOS. Интерпретатор этой версии Бейсика имеется на всех персональных компьютерах IBM PC в качестве стандартной компоненты операци­онной системы MS DOS.

**Графический Бейсик** под именем язык **Visual Basic** был создан фирмой Microsoft в качестве базового средства программирования для новейших моделей компьютеров IBM PC с операционной систе­мой Windows. Этот язык может использоваться только в среде Windows и только на старших моделях IBM PC.

Пример программы на традиционном языке Бейсик с коммента­риями, в которых записан реализованный в ней алгоритм.

**Программа Алгоритм**

**10 ' поздравление ' *алг «поздравление»***

**20 сls ' *нач***

**30 nm$ = «Оля» ' *пт$* = *«Оля»***

**40 dn$ = «с днем рождения» ' *dn$ = «с днем рождения»***

**50 print «Дорогая» + nm$ ' *вывод «Дорогая» + пт$***

**60 print «Поздравляю тебя» ' *вывод «Поздравляю тебя»***

**70 print dn$ *' вывод dn$***

**80 print «Желаю счастья» ' *вывод «Желаю счастья»***

**90 print «Твой папа» *' вывод «Твой папа»***

**100 end *' кон***

**Программы на Бейсике** состоят из операторов и комментариев. Каждый оператор соответствует некоторой операции, которую может выполнить компьютер. Комментарии включаются в тексты программ для их документирования.

Та же самая программа на структурном Бейсике:

**Программа Алгоритм**

**' поздравление ' *алг «поздравление»***

**сls ' *нач***

**nm$ = «Оля» *' пт$ = «Оля»***

**dn$ = «с днем рождения» *' dn$* = *«с днем рождения»***

**print «Дорогая» + nm$ ' *вывод «Дорогая» + пт$***

**print «Поздравляю тебя» ' *вывод «Поздравляю тебя»***

**print dn$ *' вывод dn$***

**print «Желаю счастья» ' *вывод «Желаю счастья»***

**print «Твой папа» *' вывод «Твой папа»***

**end *' кон***

Результатом выполнения на компьютере и той и другой программы будет появление на экране одного и того же текста:

**Дорогая Оля**

**Поздравляю тебя**

**с днем рождения**

**Желаю счастья.**

**Твой папа.**

В **системе программирования QBasic** на IBM PC программы могут записываться в обоих формах - с нумерацией и без нумерации строк. В версиях Бейсика для ЭВМ, не имеющих операционных систем, строки должны быть пронумерованы.

Основными **свойствами** программ для ЭВМ как одной из форм описания и разновидностей машинных алгоритмов является их выполнимость, мобильность, эффективность и правильность.

**Выполнимость** программ - возможность их выполнения на дан­ном типе компьютеров. Возможность выполнения зависит от типа ЭВМ, наличия внешних устройств, надлежащего объема оперативной и внешней памяти, операционной системы и системы программиро­вания.

**Мобильность** программ - возможность переноса программы на другой тип ЭВМ. Примером мобильности является возможность выполнения в системе структурного программирования Qbasic про­грамм, записанных на традиционном Бейсике.

**Эффективность** программ - обычно это минимальность времени их выполнения на ЭВМ. Однако, если созданные программы содержат ошибки, то утверждения об их эффективности не имеют никакого смысла.

**Правильность программ** - правильность результатов, получаемых с их помощью.

**Правильность** результатов определяется соответствием докумен­тации или другими описаниями программ.

**Программы содержат ошибки,** если их выполнение на ЭВМ при­водит к возникновению отказов, сбоев или неправильных резуль­татов. От использования программ, содержащих ошибки, следует отказываться.

**Основные типы** операторов языка Бейсик:

- операторы ввода-вывода;

- графические операторы;

- присваивания;

- обращения к функциям;

- описания данных;

- управляющие операторы;

- обращения к подпрограммам.

Примеры **операторов ввода-вывода** на экран.

Оператор Действие

**print «привет» *вывод («привет»)***

**print «корень=»; х *вывод («корень* =», *х)***

**input «a=»; а *запрос («а=», а)***

**input n *ввод (п)***

**locate st, ps *позиция (st,ps)***

Примеры **графических операторов**:

Оператор Действие

**pset(x,y),c *точка(х,у),с***

**line(x,y)-(u,v),c *линия(х,у)-(и, v), с***

**line(x,y)-(u,v),c,b *рамка(х,у)-(и,у),с***

**circle(x,y),r,c *окружность(х,у),* *r,с***

**circle(x,y),r,c,al,a2 *дуга(х,у),* *r,с,а1,а2***

**paint(x,y),c *закраска(х,у),с***

**сls *очистка\_экрана***

**screen 0,0 *текстовый\_экран***

**screen 1,0 *графический\_экран1***

**screen 2,0 *графический\_экран2***

Примеры **операторов присваивания**.

Присваивания Действие Результат

**а = 0 а := 0 а = 0**

**b = а + 1 b := a + 1 b = а + 1 = 1**

**с = 2\*b + 3 с := 2b + 3 с = 2 b + 3 = 5**

**d = b/c d := b/c d = -b/c = 0.2**

**b = b + 1 b := b + 1 b' = b + 1 = 2**

**b = b + 1 b := b + 1 b" = b' + 1 = 3**

Математические функции с примерами обращения.

Функция Смысл Пример Результат

**rnd** - случайное число от 0 до 1 **rnd**

**int (x)** - целая часть числа х **int (5/3)** 1

**abs (x)** - абсолютное значение числа **abs (-2)** 2

**sqr (x)** - квадратный корень числа **sqr (16)** 4

**sin (x)** - синус **sin (0)** 0

**cos (x)** - косинус **cos (0)** 1

**tan (x)** - тангенс **tan (0)** 0

**atn (x)** - арктангенс **atn (0)** 0

**exp (x)** - экспонента **ехр (0)** 1

**log (x)** - логарифм натуральный **log (1)** 0

К числу управляющих операторов можно отнести условные опе­раторы, имеющие следующие форму записи и смысл:

Условный оператор: Действия ЭВМ:

**if <условие> then <оператор>** ***если <условие> то <действие>***

где <**оператор**> **-** это один или несколько операторов, разделяемых двоеточием, а <**условие**> **-** это некоторое логическое условие, при соблюдении которого будут выполняться указанные операторы.

Примеры **записи условии** - простых и сложносоставных:

Условие:Запись:

**х = у х = у**

**х *≠* у х <> у**

**х > у х > у**

**х < у х < у**

**х ≤ у х <= у**

**х ≥ у х >= у**

**не (х = 1) not (x = 1)**

**(х > 0) и (у > 0) (х > 0) and (у > 0)**

**(а = 0) или (b = 0) (а = 0) or (b = 0)**

Простейшим примером программы с условными операторами является реализация алгоритма «выбор из меню»:

Сценарий «**Выбор из меню**»

***Меню: <результат >:***

***1. Новый год 1 января***

***2. День рождения 1 декабря***

***3. День знаний 1 сентября***

***выбор=?* <*n*>**

***<результат >***

Алгоритм и программа выбора по меню, соответствующие этому сценарию:

**Алгоритм Программа**

***алг «выбор по меню»*  ' выбор по меню**

***нач* cls**

***вывод («Меню»)* print «Меню:»**

***вывод («I. Новый год»)* print («1. Новый год»)**

***вывод («2. День рождения»)* print («1*.* День рождения»)**

***вывод («З. День знаний») print* («3. День знаний»)**

***запрос («выбор=», п)* input «выбор=», n**

***если п = 1 то* if n *=* I then**

***вывод («1 января»)* print «1 января»**

***если п = 2 то* if n = 2 then**

***вывод («1 декабря»)* print «1 декабря»**

***если п* = *3 то* if n = 3 then**

***вывод («1 сентября»)* print «1 сентября»**

***кон* end**

**Правильность** диалоговых алгоритмов и программ можно оценить сопоставлением их со сценарием диалога. Любое отклонение резуль­татов выполнения алгоритмов и программ от сценария диалога - это ошибка. Диалоговый алгоритм - правильный, если результаты их выполнения строго соответствуют сценарию.

**Сравнение** текста программы с описанием алгоритма, а затем ал­горитма со сценарием диалога подтверждает полное соответствие программы заданному сценарию «выбор по меню». Таким образом, правильность программ может проверяться через правильность реа­лизованных в них алгоритмов.

**В о п р о с ы**

1. Что такое программа?

2. Что такое язык программирования?

3. Каковы основные свойства программ?

4. Какие есть графические операторы?

5. Какие есть операторы ввода- вывода?

6. Какие есть математические функции?

7. Как записываются логические условия?

**З а д а ч и**

1. Составьте сценарий, алгоритм и программу с выбором из меню:

а) поздравления с Новым годом;

б) поздравления с Днем рождения;

в) регистрации даты рождения;

г) регистрации фамилии и имени.

2. Составьте сценарий, алгоритм и программу для следующих вычис­лений с выбором из меню:

а) расчета сдачи за товар;

б) расчета остатка от прибыли;

в) пересчета рубль/доллар;

г) расчета остатка времени до 18.00.

3. Составьте сценарий, алгоритм и программу рисования с выбором из меню изображений:

а) российского флага; г) украинского флага;

б) шведского флага; д) французского флага;

в) японского флага; е) британского флага.

4. Составьте сценарий, алгоритм и программу с выбором из меню следующих вычислений:

а) времени движения по длине пути и скорости;

б) длины пути по времени и скорости движения;

в) средней скорости по времени и длине пути.

5. Составьте сценарий, алгоритм и программу рисования следующих изображений с выбором из меню:

а) домика; г) автомобиля;

б) дерева; д) цветка;

в) рыбы; е) птицы.

## 4.3. Основы структурного программирования

**Алгоритмизация** - это составление алгоритмов для последующей реализации в виде программ для ЭВМ. Знание и использование сис­тематических методов превращают алгоритмизацию - в строгую дисциплину, позволяющую составлять программы на ЭВМ без ошибок.

**Порядок составления** программ:

**задача ⎯**

**алгоритмы**

**программа**

**ЭВМ**

На практике широко используются **два подхода к алгоритмизации:**

1) традиционный подход (с использованием блок-схем);

2) структурный подход (с использованием структурной записи);

**Традиционный подход** к составлению алгоритмов с применением блок-схем грешит большим числом ошибок в программах из-за их громоздкости и запутанности. Из-за этого традиционный подход к составлению программ чреват большим числом ошибок в создава­емых программах.

**Структурный подход** к программированию заключается в обяза­тельном предварительном составлении структурированных алгорит­мов с записью их на псевдокоде. Простота чтения, понимания и исправления структурированных описаний позволяет существенно уменьшить количество ошибок в алгоритмах и программах и сокра­тить время их отладки на ЭВМ.

При структурном подходе к составлению алгоритмов и программ используются три основных **правила композиции:**

1) альтернативный выбор;

2) циклический повтор;

3) вспомогательные алгоритмы (подпрограммы).

**Структурированными** считаются алгоритмы и программы состав­ленными только с использованием указанных трех правил структур­ной композиции. Неструктурированными считаются алгоритмы и программы, в которых используются операторы **goto ...** или отсутст­вует ступенчатая запись циклов и альтернатив.

Основные правила структурной композиции алгоритмов с при­мерами записи их на языке структурированного Бейсика:

**1. Альтернативный выбор:**

Алгоритм Запись

***если х > 0 то*****if х > 0 then**

***у := х*****у = х**

***иначе*** **else**

***у := -х*****у = -х**

***кесли*****end if**

**2. Циклический повтор:**

Алгоритм Запись

***пока х > 1 цикл*****do while х > 1**

***х:******= х/2*** **х = х/2**

***кцикл* loop**

3*.* **Вспомогательные алгоритмы** (подпрограммы).

Алгоритм Подпрограмма

***алг «у = |х|»*****mod: 'у = |х|**

***нач* '**

***если х > 0 то* if х > 0 then**

***у := х* у = х**

***иначе* else**

***у := -х* у = -х**

***все* end if**

***кон* return**

Обращение к алгоритму Обращение к подпрограмме

***«у = |х|»*****gosub mod**

В качестве иллюстрации приведем пример структурированного алгоритма «Галерея картинок» и соответствующей структурирован­ной программы:

Сценарий «**Галерея картинок**»

Список картинок:

1. треугольник

2. прямоугольник

3. кольцо

номер =? «N»

n = 1 n =2 n = 3

В соответствии с этими четырьмя картинками построим три вспо­могательных алгоритма рисования отдельных картинок из «Галереи» и общий алгоритм выбора картинок в соответствии с приведенным выше сценарием:

***алг «Галерея картинок»***

***нач алг «рисунок\_треугольника»***

***вывод («Список картинок:») нач***

***вывод («1. треугольник») линия (150,50)-(100,100)***

***вывод («2. прямоугольник») линия (150,50)-(200,100)***

***вывод («3. кольцо») линия (100,100)-(200,100)***

***запрос(«номер =», n) кон***

***графический\_экран***

***если n = 1 то алг «рисунок\_прямоугольника»***

***рисунок\_треугольника нач***

***инес n = 2 то рамка (50,50)-( 150,100)***

***рисунок\_прямоугольника кон***

***инес n = 3 то***

***рисунок\_кольца алг «рисунок\_кольца»***

***иначе нач***

***вывод («нет такого рисунка») окружность (100,100), 20***

***все окружность (100,100),50***

***кон кон***

Реализация данного алгоритма в виде структурированной про­граммы:

**Алгоритмы: Программа:**

***алг «Галерея картинок»*** **'Галерея картинок**

***нач*** **cls**

***вывод («Список картинок:»)*** **print «Список картинок:»**

***вывод («1. треугольник»)***  **print «1. треугольник»**

***вывод («2. прямоугольник»)*** **print «2. прямоугольник»**

***вывод («З. кольцо»)*** **print «3. кольцо»**

***запрос(«номер =», n)***  **input «номер =», n**

***если n = 1 то***  **if n = 1 then**

***рисунок\_треугольника*** **gosub treug**

***инеc n = 2 то***  **if n = 2 then**

***рисунок\_прямоугольника*** **gosub box**

***инеc n = 3 то*** **if n = 3 then**

***рисунок\_кольца*** **gosub ring**

***инеc п < 1 или n > 3 то*** **if n < 1 or n >3 then**

***вывод («нет такого рисунка»)*** **print «нет такого рисунка»**

***все*** **'все**

***кон***  **end**

***алг «рисунок треугольника»*** **treug: 'рисунок треугольника**

***нач*** **cls**

***графический\_экран*** **screen 2,0**

***линия (150,50)-( 100,100)*** **line (150,50)-(100,100),3**

***линия (150,50)-(200,100)*** **line (150,50)-(200,100),3**

***линия (100,100)-(200,100)*** **line (100,100)-(200,100),3**

***кон*** **return**

***алг «рисунок прямоугольника»*** **box: 'рисунок прямоугольника**

***нач*** **cls**

***графический\_экран*** **screen 2,0**

***рамка (50,50)-(150,100)*** **line (50,50)-(150,100),3,b**

***кон*** **return**

***алг «рисунок кольца»*** **ring: 'рисунок кольца**

***нач*** **cls**

***графический\_экран***  **screen 2,0**

***окружность (100,100),20*** **circle (100,100),20**

***окружность (100,100),50*** **circle (100,100),50**

***кон*** **return**

Данный подход - составление структурированных алгоритмов может применяться к составлению структурированных программ для любых ЭВМ на любых языках программирования - Паскаль, Си, Ада, Модула и т. д.

На практике используется более широкий набор правил струк­турной композиции алгоритмов и программ, принятых в современ­ных языках программирования, ~ правила альтернативного выбора, а также циклы с выходами и со счетчиками.

1. Условные действия.

***если у < 0 то* if у < 0 then**

***вывод («недопустим»)* print «недопустим»**

***кесли* end if**

2. Многоальтернативный выбор.

***если х > 1 то* if х > 1 then**

***у: = 1* у = 1**

***инес х < -1 то* elseif х < -1 then**

***у: = -1* у = -1**

***иначе* else**

***у:* = *х* у = х**

***кесли* end if**

3. Циклы со счетчиком:

***от k = 1 до п цикл* for k *=* 1 to n**

***вывод (k⋅k)* print k\*k**

***кцикл* next k**

4. Циклы с выходами.

***цикл* do**

***s: = s + x***  **s = s + x**

***при х < 1 выход* if х *<* 1 then exit do**

*х:* = *x/2* **x = x/2**

***кцикл* loop**

В циклах в общем случае возможны несколько выходов. Допол­нительные выходы считаются допустимыми даже для циклов со счет­чиками. Приведем примеры решения задач с использованием до­полнительных правил структурирования алгоритмов и программ.

Пример записи структурированных алгоритмов и программ с использованием циклов для алгоритма игры-эксперимента «звезд­ное небо»:

**Алгоритм Программа**

***алг «звездное небо»* ' звездное небо»**

***нач* сls**

***цикл* do**

***запрос(«звезд=», п)* input «звезд=», n**

***при п* <= *0 выход* if n <= 0 then exit do**

***графический\_экран* screen 2,10**

***от k = 1 до п цикл* for k = 1 to n**

***х: = случайное [0:200]* х = rnd\*200**

***у: = случайное [0:200]* у = rnd\*200**

***точка (х,у)* pset (x,y),3**

***кцикл* next k**

***кцикл* end do**

***кон* end**

Пример структурированного алгоритма и программы с приме­нением многоальтернативного выбора и циклов с несколькими выходами:

**Алгоритм Программа**

***алг «угадай-ка»* ' угадай-ка**

***нач* cls**

***вывод («Угадай-ка число»)* print «Угадай-ка число»**

***вывод («от 1 до 100»)* print от 1 до 100»**

***z: = случайное [0:100]* z = int (rnd\*100)**

***цикл* do**

***запрос («число =», х)*** **input «число =», х**

***при х = z вых*** **if х = z then exit do**

***если х < z то*** **if х < z then**

***вывод («мало»)*** **print «мало»**

***инеc х > z тo*** **elseif х > z then**

***вывод («много»)*** **print «много»**

***все***  **end if**

***кцикл***  **end do**

***вывод («молодец, умница»)***  **print «молодец, умница»**

***кон*** **end**

**В о п р о с ы**

1. Что такое алгоритмизация?

2. Что такое структурированные алгоритмы?

3. Что такое неструктурированные алгоритмы?

4. В чем достоинства структурированных программ?

5. В чем недостатки неструктурированных программ?

6. Можно ли гарантировать отсутствие ошибок в программах?

**З а д а ч и**

1. Постройте вспомогательные алгоритмы и подпрограммы с выде­лением параметров для рисования следующих блоков:

а) крыша;

б) дерево;

в) стена с окном;

г) столб.

2. Предложите рисунки и составьте алгоритмы рисования, используя вспомогательные алгоритмы из предыдущего задания, для следующих строений:

а) домика с окном и деревом;

б) домика с двумя окнами;

в) домика с собачьей будкой;

г) двухэтажного домика,

3. Составьте алгоритм вывода на экран полной таблицы умножения.

4. Составьте, используя вспомогательные алгоритмы из предыдущих задач, алгоритмы изображения на экране:

а) многосекционных домов с различным числом секций;

б) многоэтажных домов с различным числом этажей и секций.

## 4.4. Основы безошибочного программирования

Основной **недостаток традиционной практики** составления про­грамм для ЭВМ заключается в том, что при таком подходе никто не может гарантировать отсутствие в них ошибок. Особенностью традиционной практики является поиск ошибок в программах при их отладке на ЭВМ.

Однако, так как **число ошибок в программах** заранее неизвестно, то неизвестна заранее и продолжительность отладки программ на ЭВМ. Более того даже после «завершения» отладки никто не может гарантировать отсутствие ошибок. Естественно, что использование таких программ, приводит к возникновению отказов, сбоев и полу­чению неверных результатов.

Структурный подход **снижает количество ошибок** в алгоритмах и программах. Однако и при этом подходе число ошибок также зара­нее неизвестно. Хотя структурная форма записи и упрощает поиск и исправление ошибок в текстах программ, гарантии отсутствия ошибок структурный подход не дает.

**Однозначные суждения** об отсутствии или наличии ошибок в алгоритмах и программах возможны только при наличии описаний конечных результатов их выполнения. Такие описания принято называть спецификациями.

**Спецификации программ** - это точные, математически строгие описания результатов выполнения алгоритмов и программ. Только при наличии спецификаций возможно создание алгоритмов и про­грамм, в которых можно гарантировать отсутствие ошибок.

Более того, при систематическом **использовании спецификаций** возможен не только анализ правильности алгоритмов и программ, но и становится возможным составление программ с одновремен­ным доказательством правильности.

**Безошибочное программирование** - это составление алгоритмов и программ с гарантиями отсутствия в них ошибок. А составление алгоритмов и программ с одновременным доказательством правиль­ности называется доказательным программированием. И в том и другом подходе необходимо составление спецификаций.

Для составления программ **на любом языке программирования** весьма полезно предварительное составление реализуемых в них алгоритмов. Эти описания алгоритмов вместе со спецификациями позволяют в полной мере оценить правильность составленных про­грамм. Пример составления алгоритмов с использованием в качестве иллюстрации спецификаций сценария диалога с ЭВМ:

Сценарий «**Галерея картинок**»

Список картинок:

1. треугольник

2. прямоугольник

3. кольцо

номер = ? <N>

n =1 n = 2 n = 3

В соответствии с этими четырьмя картинками построим три вспо­могательных алгоритма рисования отдельных картинок из «Галереи» и общий алгоритм выбора картинок в соответствии с принятым сценарием:

***алг «Галерея картинок»***

***нач* алг «рисуиок\_треугольника»**

***вывод («Список картинок:»)* нач**

***вывод («1. треугольник»)* линия(150,50)-(100,100)**

***вывод («2. прямоугольник»)* линия(150,50)-(200,100)**

***вывод («3. кольцо»)* линия(100,100)-(200,100)**

***запрос («номер=», п)* кон**

***графический\_экран***

***если п = 1 то* алг «рисунок\_прямоугольника»**

***рисунок\_треугольника* нач**

***инес п* = *2 то* рамка(50,50)-(150,100)**

***рисунок\_прямоугольника* кон**

***инес п = 3 то***

***рисунок\_кольиа*** **алг «рисунок\_кольца»**

***иначе*** **нач**

***вывод («нет такого рисунка»)* окружность( 100,100),20**

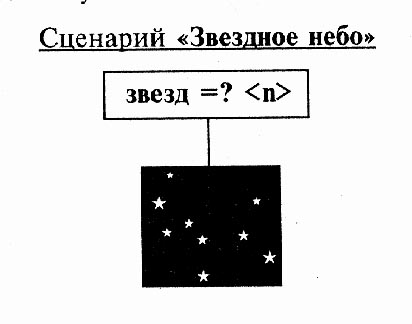
***все* окружность(100,100),50**

***кон* кон**

**Правильность** каждого из вспомогательных алгоритмов и подпро­грамм определяется сравнением с соответствующими фрагментами сценария, а правильность всего алгоритма и соответствующей про­граммы - со сценарием в целом.

Данный подход к составлению алгоритмов и программ с исполь­зованием спецификаций - позволяет реализовать основную идею безошибочного программирования - создание алгоритмов и про­грамм, **правильных по построению.** Такой подход может применяться к составлению алгоритмов и программ для любых современных языков программирования - Паскаль, Си, Ада, Модула, Бейсик и т. д.

Приведем примеры составления сложных алгоритмов и программ с циклами с использованием спецификаций. Первый пример - построение алгоритма и программы изображения на экране картинки «Звездное небо» из n случайных точек:



В приводимом ниже алгоритме для формирования и вывода по­следовательности случайных точек на экране используется цикл со счетчиком и датчик случайных чисел для генерации координат «звезд».

**Алгоритм Программа**

***алг «звездное небо»* ' звездное небо**

***нач* сls**

***запрос(«звезд=», п)* input «звезд=», n**

***графический\_экран* screen 2,0**

***от k* = *1* *до п цикл* for k = 1 to n**

***x:* = *случайное [0:200]* х = rnd\*200**

***у: = случайное [0:200]* у = rnd\*200**

***точка (х,у)* pset (x,y),3**

***кцикл* next k**

***кон* end**

Второй пример - составление с использованием спецификаций алгоритма и программы игры «Угадай-ка». В этой игре ЭВМ «зага­дывает» число от 0 до 100, а человек должен его отгадать, вводя пробные числа с клавиатуры. Для составления алгоритма и програм­мы примем следующий сценарий:

Сценарий **«Угадай-ка»**

|  |  |
| --- | --- |
| *Угадай число от 0 до 100* |  |
| *число = ? < х>* | *\** |
| *мало* |  |
| *много* |  |
| *молодец, умница* |  |

Для реализации этого сценария воспользуемся циклом с выхо­дом, в котором задается вопрос **число=?** и проверяются числа, вво­димые человеком. Выход из цикла происходит после совпадения ответа с числом, задуманным ЭВМ:

**Алгоритм Программа**

***алг «угадай-ка»* ' угадай-ка**

***нач* сls**

***вывод («Угадай число»)* print «Угадай число»**

***вывод («от 1 до 100»)* print «от 1 до 100»**

**z: = *случайное [0:100]* z = int (rnd\* 100)**

***цикл* do**

***запрос( «число=», х)* input «число=», х**

***при х = z вых* if х = z then exit do**

***если х <* z *то*  if х < z then**

***вывод («мало»)*  print «мало»**

***инеc х > z то* elseif х > z then**

***вывод («много»)* print «много»**

***все* end if**

***кцикл* loop**

***вывод («молодец, умница»)* print «молодец, умница»**

***кон*  end**

Сравнение алгоритма со сценарием показывает их полное соот­ветствие друг другу.

**В о п р о с ы**

1. Сколько ошибок содержится в программах?

2. Как долго длится отладка программ?

3. Что такое спецификации программ?

4. Зачем нужны спецификации?

5. Можно ли гарантировать отсутствие ошибок в программах?

6. Что такое систематический подход к алгоритмизации?

**З а д а ч и**

1. Составьте сценарий и алгоритм диалога «Распорядок дня», с по­мощью которого можно узнать, что запланировано на заданный час дня.

2. Составьте сценарий и алгоритм диалога с выбором по меню;

а) национальных флагов;

б) каталога строительных блоков;

в) набора рисунков;

г) каталога строений.

3. Предложите сценарии и алгоритмы рисования на экране абстракт­ных рисунков:

а) из случайных разноцветных точек;

б) из случайных разноцветных отрезков;

в) из случайных разноцветных рамок;

г) из случайных разноцветных окружностей;

д) из случайных разноцветных кругов;

е) из случайных разноцветных окошек.

4. Составьте сценарий и алгоритм, моделирующий на экране бро­уновское движение частиц.

## 4.5. Средства обработки данных

**Автоматизированная обработка данных** - одна из основных массовых проблем, решаемых с помощью ЭВМ. На персональных компьютерах IBM PC базовым средством обработки данных является язык программирования Basic. В операционной системе Windows это язык считается основным языком разработки программ для компьютеров IBM PC.

Основной особенностью языков структурного и графического программирования Бейсика как языка обработки данных являются операторы данных **data**, позволяющие описывать данные непосредст­венно в текстах программ. Пример и реализация алгоритма обработки данных:

***алг «день рождения» '* день рождения**

***нач* cls**

***вывод («день рождения»)* print «день рождения:»**

***чтение пт$, dn, ms, gd* read nm$, dn, ins, gd**

***вывод nm$; dn; ms; gd* print nm$; dn; ms; gd**

***кон* end**

***дано: Саша, 18, 10, 1980* data «Саша», 18,10,1980**

Выполнение программы на компьютере приведет к появлению на экране следующих строк:

**день рождения:**

**Саша 18 10 1980**

Для решения этой задачи для других данных необходимо внести изменения в оператор данных **data** и вновь запустить программу на выполнение. Пример изменения данных:

**дано: *Оля, 1, 12, 1974* data *«Оля», 1,12,1974***

В традиционных версиях языка Бейсик с нумерацией строк опе­раторы **data** выделяются в отдельные группы и нумеруются обычно с числа 1000. Это позволяет четко отделить в программах описание данных от операторов их обработки:

***алг «дни рождения»* 10 ' дни рождения**

***нач* 20 cls**

***вывод («день рождения:»)* 30 print «день рождения:»**

***чтение nт$, dn, ms, gd* 40 read nm$, dn, ms, gd**

***вывод nm$; dn; ms; gd* 50 print nm$; dn; ms; gd**

***кон* 60 end**

***дано: Иванов, Саша, 18,10,1980* 1000 data «Саша», 18,10,1980**

При размещении нескольких таблиц или других групп данных в программах на Бейсике полезным средством являются операторы restore (операторы чтения данных с заданного номера или метки):

1) оператор чтения данных после метки test:

**restore test** - *чтение данных после метки* **test**;

2) оператор чтения данных с оператора 1000:

**restore 1000** - *чтение данных, начиная с 1000-го оператора;*

3) оператор чтения данных с самого начала:

**restore** - *чтение данных сначала.*

В задачах обработки данных переработке подвергаются не только числовые данные, но и символьная информация. Для этих целей в программах используются символьные данные, переменные и мас­сивы.

**Символьные данные** - это последовательности символов. В текстах программ на Бейсике символьные данные заключаются в двойные кавычки. Примеры: «мама», «корень=», «2 + 1» и т.д. Во входных данных символьные данные записываются в соответствии с входными спецификациями.

**Символьные переменные** - это переменные, значениями которых являются символьные данные. В программах на Бейсике символь­ными явлются те переменные, к имени которых справа приписан знак $. Примеры символьных переменных: **s$, p$, sl$, pr$.**

**Числовые данные** и переменные в языке Бейсик могут быть трех основных типов - целочисленные, вещественные и вещественные двойной точности. В программах для этих типов переменных ис­пользуются следующие обозначения:

**n%, m%, nl%, m3%** - *целочисленные*

**х, у, xl, y5** - *вещественные*

**а#, b#, al#, b8#** - *вещественные двойной точности*

В качестве примера решения задач обработки данных рассмот­рим алгоритм и программу вывода списка дней рождения членов семьи по данным, представленным в следующей таблице:

Дни рождения:

|  |  |  |  |
| --- | --- | --- | --- |
| **Мама** | **26** | **6** | **1949** |
| **Папа** | **22** | **5** | **1946** |
| **Сережа** | **25** | **10** | **1973** |
| **Оля** | **1** | **12** | **1974** |

Для представления данных из этой таблицы в программе восполь­зуемся следующей последовательностью операторов **data:**

Дни рождения:

|  |  |  |  |
| --- | --- | --- | --- |
| **Мама** | **26** | **6** | **1949** |
| **Папа** | **22** | **5** | **1946** |
| **Сережа** | **25** | **10** | **1973** |
| **Оля** | **1** | **12** | **1974** |

**dni: ' дни рождения**

**data «мама», 26, 6, 1949**

**data «папа», 22,5, 1946**

**data «Сережа», 25, 10, 1973**

**data «Оля», 1, 12, 1974**

**data «», 0, 0, 0**

**Обратите внимание!**

1. Каждый оператор **data** здесь отвечает одной строке таблицы.

2. Последний оператор **data** содержит пустую «запись» - пустое имя «» и три нуля, означающие конец данных.

Такая **форма представления** данных позволяет достаточно просто вносить изменения, исправления и добавления в данные. Эти изме­нения в таблице переносятся в соответствующие операторы **data,** а добавление или удаление строк в таблице отображается добавлением или удалением соответствующих операторов в программе.

Рассмотрим алгоритм и программу вывода списка дней рождения в семье, составленные в соответствии с выбранным представлением данных:

***алг «дни рождения» '* дни рождения**

***нач* сls**

***вывод («дни рождения»)* print «дни рождения»**

***чтение таблицы dni* restore dni**

***цикл* do**

***чтение (пп, d, т, g)* read nn$, d, m*,* g**

***при пп* = *«» вых* if nn$ = «» exit then do**

***вывод (пп, d, m, g)* print nn$, d, m, g**

***кцикл* loop**

***кон* end**

Для формирования и обработки новых групп данных в программах используются массивы. **Массив** в программе - это область опе­ративной памяти ЭВМ, используемая для размещения некоторой совокупности данных.

Использование массивов в программах на Бейсике требует опи­сания их с помощью **операторов dim.** В операторах **dim** для каждого массива указывается его имя и размеры. Массивы в программах могут быть одномерными, двумерными, трехмерными и т. д.

Примеры описаний массивов:

одномерные массивы из 20 элементов -

**dim nm$(20), d(20), m(20)**

двумерные массивы из 2х10 и 10х10 элементов –

**dim fm$(2,10), tb(10,10)**

Обращения к элементам массивов записываются в зависимости от размерности, указанной в их описаниях. Примеры обращений к одномерным и двумерным массивам:

**nm$(4) = «Костя»**

**d(4) = 10**

**fm$(l,10) = «Петров»**

**tb(3,4) = 3\*4**

В программах на Бейсике операторы **dim** являются выполняемы­ми. Результатом их выполнения является выделение участков памя­ти для хранения соответствующих массивов. По этой причине в ка­честве размеров массивов могут указываться переменные, которые должны получить конкретные положительные значения до выпол­нения оператора **dim.**

Описание двумерного массива с переменной **n** в качестве его раз­меров:

**n = 5 ' n = 5**

**dim tb(n,n) , ' массив tb[1:n, 1:n]**

В качестве примера использования массивов с переменными раз­мерами приведем алгоритм и программу формирования «Таблицы умножения n×n».

Таблица умножения

1 2 3 4 5

2 4 6 8 10

3 6 9 12 15

4 8 12 16 20

5 10 15 20 25

В приведенных ниже алгоритме и программе расчета и вывода таблицы умножения для ее размещения используется двумерный массив tb(n, n) c n = 5:

***алг «таблица умножения»* ' таблица умножения**

***п*=5 n=5**

***массив tb[1:n, 1:n]* dim tb(n,n)**

***нач* сls**

***от k* = *1* *до п цикл* for k = 1 to n**

***от 1* = *1* *до п цикл* for l = 1 to п**

***tb[k,l]:* = *k\*l* tb(k,l) = k\*l**

***вывод tb[k,l]* print tb(k,l);**

***кцикл* next 1**

***нов\_строка* print**

***кцикл* next k**

***кон* end**

Запуск этой программы на ЭВМ приведет к получению приве­денной выше картинки с таблицей умножения размера 5х5. Для получения таблицы умножения размера 8х8 или 10 х 10 достаточно изменить в программе значение n *=*5 на n = 8 или n = 10.

Перечисленных **базовых средств достаточно** для решения большого числа задач обработки данных: экономических, статистических, инженерных, научных и т.п. Однако при постановке решения задач обработки данных важно четко различать место размещения и виды обрабатываемых данных.

По способу использования при решении задач различаются сле­дующие данные:

исходные;

результирующие.

**Исходные данные** - конкретные данные решаемых задач, отвеча­ющие принятой постановке. Исходные данные могут оказаться как допустимыми, так и недопустимыми по постановке решаемых задач.

**Результирующие данные** - это результаты решения поставленных задач при введенных исходных данных. Сообщения о невозмож­ности решения задачи также считаются результирующими данными.

По **способу размещения** и использования в обрабатывающих алгоритмах и программах данные подразделяются на:

* входные;
* выходные;
* сохраняемые.

**Входные данные** - это данные, вводимые в ЭВМ во время работы программы. Входные данные могут вводиться с клавиатуры, магнит­ных дисков или с помощью других устройств ввода информации.

**Выходные данные** - данные, выводимые ЭВМ как результат ра­боты программ. Выходные данные могут выводиться на экран, на печать, на магнитные диски или другой носитель информации.

**Сохраняемые данные** - данные, которые хранятся в долговремен­ной памяти ЭВМ и могут обновляться как результат работы про­грамм. Эти данные могут храниться и многократно обновляться на магнитных дисках в течении длительного промежутка времени.

В качестве примера рассмотрим задачу поиска номеров телефо­нов по телефонному справочнику. Исходной информацией в этой задаче является «Телефонный справочник», который можно пред­ставить следующей таблицей:

Телефонный справочник

|  |  |
| --- | --- |
| Вова | 125-14-70 |
| Саша | 222-01-02 |
| Маша | 102-99-00 |

Результирующая информация - номера телефонов и сообщения об отсутствии таких сведений. Информация о результатах поиска информации может выводиться на экран ЭВМ. Диалог с компьюте­ром может проходить по следующему сценарию, в котором отража­ются исходные и выходные данные:

Сценарий:

|  |
| --- |
| поиск номера телефона  имя = ? <имя> |
| телефон: <номер> |
| нет такого |

Для хранения таблицы «Телефонного справочника» в программе можно воспользоваться следующими операторами **data**:

**tel: 'номера телефонов:**

**data «Вова», «125-14-80»**

**data «Саша», «222-01 -02»**

**data «Маша», «102-99-00»**

**data «», «»**

При выбранных представлении данных и сценарии диалога решением могут служить следующие алгоритм и программа:

**Алгоритм** **Программа**

***алг «Телефонный справочник» '* Телефонный справочник**

***нач* сls**

***вывод («поиск номера телефона»)* print «поиск номера телефона»**

***запрос(«имя=», NN)* input «имя=», NN$**

***чтение-таблицы tel* restore tel**

***цикл* do**

***чтение (имя, пот)* read im$, nm$**

***если имя* = *NN то* if im$ = NN$ then**

***вывод («номер:»,пот)* print «номер:»,nm$**

***выход [из цикла]* exit do**

***инес имя* = *«» то* elseif im$ = «» then**

***вывод («нет такого»)* print «нет такого»**

***выход [из цикла]* exit do**

***все* end if**

***кцикл* loop**

***кон* end**

Из приведенного примера видно, что при составлении алгорит­мов и программ обработки данных важную роль играют не только сценарии ввода-вывода данных в ЭВМ, но и **представление** **данных**. От выбора этих представлений существенно зависят способы доступа к данным и процедуры обработки.

Однако наиболее важным при составлении алгоритмов и программ обработки данных прежде всего является четкое **определение исход­ных и результирующих данных**, а уже затем - подбор представлений входных, выходных и сохраняемых данных на ЭВМ.

**Систематические методы** разработки алгоритмов и программ обработки данных состоят в том, что постановка решаемых задач, выбор представлений данных и составление спецификаций диалога проводятся до составления детальных алгоритмов и программ обработки данных.

Подобный подход к составлению алгоритмов и программ обра­ботки данных позволяет проверять правильность составляемых алгоритмов и программ по отношению к этим спецификациям и обеспечить в них **полное устранение ошибок**.

Приведем пример систематического составления алгоритмов и программ обработки данных с использованием спецификаций для решения задачи «Выбор друзей по росту». Допустим, что исходные данные этой задачи представлены следующей таблицей:

|  |  |  |
| --- | --- | --- |
| **фамилия** | **имя** | **рост** |
| Иванов | Саша | 180 |
| Петров | Вова | 160 |
| Сидоров | Миша | 190 |

Примем, что запросы на поиск друзей по росту и результаты по­иска будут выводиться на экран по следующему сценарию:

Сценарий «**Поиск друзей**»

|  |
| --- |
| выбор друзей по росту  мин\_рост = ? <min>  макс\_рост = ? <max> |
| <фамилия> <имя> |
| нет таких |

Для представления данных о друзьях в программе воспользуемся следующими операторами **data**:

**dan: 'данные о друзьях**

**data «Иванов», «Саша», 180**

**data «Петров», «Вова», 160**

**data «Сидоров», «Миша», 190**

**data «», «», 0**

Тогда в качестве решения на ЭВМ поставленной задачи в соот­ветствии с выбранными сценарием и представлением сохраняемых данных, могут быть приняты следующие алгоритм и программа обработки данных.

**Алгоритм** **Программа**

***алг «выбор друзей»* ' выбор друзей**

***нач* сls**

***вывод («выбор друзей по росту»)* print «выбор друзей по росту»**

***запрос («мин\_рост =>», min)* input «мин\_рост =>», mn**

***запрос («макс\_рост =<», тах)* input «макс\_рост =<», mх**

***чтение-таблицы dan* restore dan**

***n: = 0* n = 0**

***цикл* do**

***чтение (фам, имя, r)* read fm$,im$,r**

***при фам = «» вых*  if fm$ = «» then exit do**

***если min* ≤ *r и r* ≤ *max то* if mn<= r and r <= mx then**

***вывод (фам, имя)* print fm$, im$**

***n*: = *n+1*  n *=* n+1**

***все* end if**

***кцикл* loop**

***если n* = *0 то* if n = 0 then**

***вывод «нет таких»* print «нет таких»**

***кон* end**

Сравнение приведенных алгоритма и программы со сценарием диалога показывает их полное соответствие друг другу. Прогон этой программы на ЭВМ при самых различных вариантов запросов под­твердит правильность ее работы, а доказательство ее правильности потребует знания техники анализа результатов ее выполнения для всех комбинаций исходных данных.

**В о п р о с ы**

1. Что такое исходные и результирующие данные?

2. Что такое входные, выходные и сохраняемые данные?

3. Что такое представление данных?

4. Как описываются массивы в программах на Бейсике?

5. Какие типы переменных есть в программах на Бейсике?

6. Как описываются данные в программах на Бейсике?

**3 а д а ч и**

1. Составьте сценарий, алгоритм и программу поиска номера теле­фона по фамилии с представлением сведений в последовательности операторов **data**.

2. Составьте сценарий, алгоритм и программу поиска по имени дней рождения родных: мамы, папы, сестер и братьев, используя операторы **data**.

3. Составьте сценарий, алгоритм и программу поиска следующих данных о друзьях, используя операторы **data** для получения сведений:

а) о росте друзей;

б) о весе друзей;

в) о цвете глаз.

4. Составьте сценарий, алгоритм и программу поиска сведений о расписании занятий по дням недели, используя операторы **data**.

5. Составьте сценарий, алгоритм и программу поиска сведений о расписании занятий, используя операторы **data**:

а) по названию предмета;

б) по дням недели;

в) по номеру урока.

6. Составьте алгоритм и программу построения изображения лома­ной по координатам точек, записанных в последовательности операторов **data**.

7. Составьте алгоритм и программу вывода изображений ткани из цветных кругов по данным об их центрах и радиусах, записанных в последовательности операторов **data**.

# Глава 5. ТЕХНОЛОГИЯ РЕШЕНИЯ ЗАДАЧ

## 5.1. Решение задач на ЭВМ

Решение задач должно начинаться с их точной постановки. **Постановка задач** - это четкое выделение того, что требуется, и того, что дано:

Постановка

**Задача**

**требуется?** **дано?**

Следующий этап - определение способа решения задачи. **Способ решения** - это набор действий, позволяющих получить требуемое из исходного:

Решение

**Задача**

**исходное → способ → результаты**

**Результат правильный,** если он отвечает требованиям. Получение результатов - главное в решении любых задач. Отсутствие или неправильность результатов говорит о неуспехе деятельности.

**Результат неправильный,** если он не соответствует требованиям. Однако при отсутствии четких требований невозможно однозначно судить о правильности или неправильности результатов.

При **решении на ЭВМ** постановка задач предполагает представле­ние требуемого и исходного в виде данных. Способы решения задач на ЭВМ в такой постановке должны быть представлены соответст­вующими алгоритмами и программами обработки данных.

Решение на ЭВМ

**Задача**

**↓**

**Программа**

**↓**

**данные → ЭВМ → результаты**

При отсутствии готовых программ для решения задач возникает **проблема создания** соответствующих алгоритмов и программ. В любом случае необходимо подобрать и определить способы, методы и сред­ства для решения поставленных задач.

**Систематический подход** к составлению программ предполагает в качесте первого этапа составление спецификаций - описаний форм ввода и хранения данных в ЭВМ, а также получения и вывода результатов. Эти спецификации в дальнейшем будут использоваться для оценки правильности созданных программ.

Для **диалоговых программ** в роли таких спецификаций выступают сценарии диалога - полные описания результатов и правил работы с ЭВМ при решении поставленных задач. Только после создания таких спецификаций должны составляться соответствующие им алгоритмы и программы.

Составление программ

**задача → способы**

**↓ ↓**

***сценарий* → *алгоритмы***

**↓ ↓**

**ЭВМ ← программа**

Приведенная схема представляет основной принцип **системати­ческих методов** составления алгоритмов и программ для решения различных прикладных задач - экономических, математических, физических, инженерных и т. д.

Особенностью систематических методов является возможность полного **устранения ошибок** из алгоритмов и программ. При этом подходе программы сверяются с описаниями алгоритмов, а алгорит­мы - с описаниями сценариев и методов решения.

Такой систематический подход к составлению алгоритмов и про­грамм может применяться к решению на ЭВМ любых прикладных задач с использованием самых **различных языков программирования** - Бейсик, Паскаль, Си и им подобные. Приведем примеры системати­ческого решения задач.

**Первая задача:** подсчет площади треугольника по длинам сторон.

a b

c

Постановка Сценарий

Дано: а, b, с - длины сторон, *площадь треугольника*

Треб.: S - площадь треугольника, *длины сторон:*

При: а > 0, b > 0, с > 0, а =? <а>

a < b +c, b < a + c, c < a + b. b =? <b>

с =? <с>

Метод решения *площадь* = <S>

S = недопустимы длины

р = (а + b + с)/2

**Обратите внимание:** в постановке задачи в исходные условия вклю­чены ситуации, когда решение может не существовать. А именно, здесь указаны три неравенства треугольника и условия положитель­ности длин сторон. При нарушении этих условий треугольника просто не существует и тем более нельзя говорить о его площади.

Для **надежности программ** такого рода ситуации (когда нет реше­ний) должны быть предусмотрены в сценарии диалога. В этих случаях в сценарий необходимо включить сообщения с диагностикой причин отказов: отсутствие решений, недопустимость данных, некоррект­ность команд, противоречивость фактов и т. п.

**Алгоритм Программа**

***алг «площадь треугольника»* ' площадь треугольника**

***нач***  **cls**

***вывод («площадь треугольника»)* ? «площадь треугольника»**

***вывод («длины сторон:»)* ? «длины сторон:»**

***запрос («а=», a)* input** **«a=», a**

***запрос («b=», b)* inpnt «b=», b**

***запрос («с=», с)* input «c=», c**

***если не (а > 0 и b > 0 и с > 0) то* if a<=0 or b<=0 or c<=0 then**

***вывод («недопустимы длины»)* ? «недопустимы длины»**

***инеc не (а < b* + с *и b < а +* elseif not (a < b+ с and b < а + с**

***+с и с<а+b)то* and с < а + b) then**

***вывод («недопустимы длины»)* ? «недопустимы длины»**

***иначе*  else**

***р := (а + b + с)/2*  р = (а+ b +с)/2**

**S *:= *  S = sqr (p\*(p-a)\*(p-b)\*(p-c))**

***вывод («площадь=», S)*  ? «площадь=», S**

***все* end if**

***кон* end**

Рассмотренный пример служит иллюстрацией постановки задачи, в которой выделены как требуемые и исходные данные, так и условия допустимости исходных данных. Такая **постановка задачи** позволяет заранее выделить все случаи и ситуации недопустимости данных, что в дальнейшем понадобится при составлении сценария диалога с компьютером.

В общем случае **математическая постановка** задач должна содер­жать не только условия допустимости данных, но и точное описание требований к результатам:

1) **дано:** перечень исходных данных;

2) **треб:** перечень требуемых данных;

3) **где:** требования к результатам;

4) **при:** условия допустимости данных.

**Вторая задача:** определение среднего арифметического последо­вательности из N чисел **х1,** **х2, ..., хN.** Приведем постановку, метод решения и сценарий диалога для решения этой задачи.

Постановка задачи Сценарий

Дано: N - количество чисел, *среднее N чисел*

x1, х2, .., хN - числа, *чисел* =? *<N>*

Треб.: s - среднее N чисел. \*

Где: s = (х1, + х2 +...+ хN )/ N. 1: <х1>

При: N > 0. 2: <х2>

………..

Метод решения N: <хN>

S0 = 0 *среднее* = <s>

Sk *=* Sk-1 + хk

[k = 1, ..., N] *недопустимо N*

s = SN / N

Обратите внимание: **метод вычисления** среднего N чисел здесь описан через подсчет суммы чисел. Правильность метода может быть проверена по отношению к требованиям постановки задачи.

Приведем алгоритм и программу обработки данных, составлен­ные в точном соответствии с выбранным сценарием и методом решения:

**Алгоритм Программа**

***алг «среднее арифметическое»* ' среднее арифметическое**

***нач* cls**

***вывод («среднее N чисел»)* ? «среднее N чисел»**

***запрос («чисел=», N)* input «чисел=», N**

***S* := *0* S = 0**

***если N <= 0 то* if N <= 0 then**

***вывод («недопустимо N»)* ? «недопустимо N»**

***инеc N > 0 то* elseif N > 0 then**

***от k = 1 до N цикл* for k = 1 to N**

***вывод (k, «:»)* ? k, «:»**

***запрос (x)* input x**

***S* := *S + x* S = S + x**

***кцикл* next k**

***s* := *S/N* s = S/N**

***вывод («среднее =», s)* ? «среднее=», s**

***все* end if**

***кон* end**

При решении сложных задач для проверки правильности состав­ляемых алгоритмов и программ обязательны не только математичес­кое описание постановки задач, но и описание выбранных методов решения.

Приведем пример разработки программы обработки данных с математической постановкой задачи и полным описанием метода решения.

**Третья задача:** определение самого легкого из учеников по данным из таблицы, содержащей N строк:

**фамилия рост вес**

|  |  |  |
| --- | --- | --- |
| Иванов | 185 | 85 |
| Петрова | 165 | 65 |
| Сидоров | 170 | 80 |

Постановка задачи Сценарий

Дано: (D1, ..., DN) - данные учеников. *Данные об учениках*

где D *=* [Fam, R,V] - состав данных, *фамилия вес*

Fam - фамилия, R - рост, V -вес

Треб.: Famm - фамилия ученика. <Fam1> <V1> \*

Где:m: Vm = Min (V1 ..., VN). … …

При: N > 0. <FаmN> <VN>

Метод решения *самый легкий:*

*Min (V1,.. Vn):* Fam m > <Vm >

*min* = *V1*

*от k* = *1* *до п цикл* **Представление данных**

*если Vk < min то* **dan: 'данные учеников:**

*min: = Vk* **data «Иванов», «Вова», 180,80**

*кцикл* **data** **«»,»»,0 ,0**

Выбранному сценарию, методу решения и представлению дан­ных соответствуют следующие алгоритм и программа на Бейсике.

**Алгоритм Программа**

***алг «самый легкий ученик»* ' самый легкий ученик**

***нач* cls**

***вывод («Данные об учениках»)* ? «Данные об учениках»**

***вывод («фамилия вес»)*  ? «фамилия вес»**

***N: = 0* n = 0**

***цикл* do**

***чтение (Fam, r, v)*  read famS, r, v**

***при Fam = «» выход*  if fam$ = «» then exit do**

***вывод (Fam, v)* ? fam$, v, r**

***N:=N+1* n = n+1**

***если N == 1 или V < Vmin то* if n=l or v < vmin then**

***Vmin: = V*  vmin = v**

***Fmin: = Fam*  fmin$ = fam$**

***все* end if**

***кцикл* loop**

***вывод («самый легкий:»)* ? «самый легкий:»**

***вывод (Fmin, Vmin)*  ? fmin$, vmin**

***кон* end**

В общем случае систематический подход к решению задач на ЭВМ требует для проверки правильности алгоритмов и программ не только математической постановки задач, но и обязательного описания выбранных методов решения.

**Систематический подход:**

**задача → способы**

↓ ↓

**постановка → методы**

↓ ↓

**сценарий → алгоритмы**

↓ ↓

**ЭВМ ← программа**

Рассмотрим пример систематического составления алгоритма и программы для решения на ЭВМ достаточно сложной задачи обра­ботки данных.

**Четвертая задача:** Определить суммы элементов столбцов в матрице Anxm:



Приведем обобщенную постановку задачи и описание соответст­вующих общего метода решения и сценария диалога.

Постановка задачи Сценарий

Дано: *Матрица* <N>×<M>

(a11 … a1N) < a11> ... < a1N >

(... ... ... ) - матрица Anxm ... ... ...

(aMl … aMN) < aMl > … < aMN >

Треб.: Суммы элементов:

(S1 ..., SN) - суммы столбцов <S1> ... *<S*N*>*

Где:

Si = аi1 + ...+ аiM

[i = (1… N)]

При: N > 0, М > 0.

**Метод вычислений Представление данных**

**sk0 = 0 matr: ' матрица Anm:**

**sk1 = ak1+ sk1-1 data** **3, 4**

**[1 = (1 ... M)] data I, 2, 3, 4**

**Sk = SkN data 0, 1, 2, 3**

**[k = (1 ... N)] data 0, 0, 1, 2**

В предлагаемой ниже программе для представления матриц ис­пользуются операторы **data.** В первом из этих операторов записаны размеры, а в каждом последующем операторе - строки матрицы:

**Алгоритм Программа**

***алг «сумма строк матрицы»* ' сумма строк матрицы**

***нач* cls**

***чтение (п, т) .* read n, m**

***если п > 0 и т > 0 то* if N > 0 and М > 0 then**

***массив А[1:п,1:т]* dim A (N,M)**

***массив S[1:n]* dim S(n)**

***ввод-вывод\_матрицы* gosub vvod 'ввод-матрицы**

***суммирование\_строк* gosub sum 'суммирование**

***от k* = *1 до п цикл* for k= 1 to n**

***выв (s[k])* ? s[k]**

***кцикл* next k**

***все* end if**

***кон* end**

***алг «суммирование строк»* sum: 'суммирование строк**

***нач* ' нач**

***от k* = *1 до N цикл* for k = 1 to n**

***s[k] := 0* s[k] = 0**

***от l* = *1 до М цикл* for I = 1 to m**

***s[k] := s[k]* + *A[k,l]* s[k] = s[k] + a[k,l]**

***кцикл* next I**

***кцикл* next k**

***кон* return**

***алг «ввод-вывод\_матрицы»* vvod: 'ввод-вывод\_матрицы**

***нач* ' нач**

***вывод («Матрица», N, «х», М)* ? «Матрица»; m; «х»; m**

***от k = 1 до N цикл* for k = 1 to n**

***от I = 1 до М цикл*  for l = 1 to m**

***чтение (A [k,l])* read A (k,l)**

***вывод (A [k,l])* ? A (k,l)**

***кцикл*  next 1**

***нов\_строка* ?**

***кцикл*  next k**

***кон* return**

**В о п р о с ы**

1. Что такое постановка задачи?

2. Что включается в постановку задач?

3. Что такое способ решения?

4. Что такое метод решения?

5. Каков порядок решения новых задач?

6. Что такое систематическая разработка алгоритмов и программ?

**З а д а ч и**

1. Приведите постановку задачи, сценарий, алгоритм и программу подсчета сумм:

а) нечетных чисел;

б) квадратов целых чисел;

в) кубов целых чисел.

2. Приведите постановку задачи, сценарий, алгоритм и программу подсчета сумм:

а) членов арифметической прогрессии;

б) членов геометрической прогрессии.

3. Для последовательности чисел х1, х2 ..., хN приведите постановку задачи, составьте сценарий, алгоритм решения и программу:

а) подсчета суммы всех чисел;

б) вычисления среднего арифметического чисел;

в) определения наибольшего из чисел;

г) определения наименьшего из чисел.

4. Для данных об учениках, содержащих сведения об их росте и весе, приведите постановку задачи, составьте сценарий, алгоритм и програм­му определения:

а) самого высокого ученика; г) самого легкого ученика;

б) самого низкого ученика; д) средний рост учеников;

в) самого тяжелого ученика; е) средний вес учеников.

5. Для данных о днях рождения своих друзей и родных приведите постановку задачи, составьте сценарий, алгоритм решения и программу:

а) определения ровесников;

б) определения людей, родившихся в один день;

в) самого молодого из своих друзей и родных;

г) самого старшего из своих родных и друзей.

## 5.2. Анализ правильности алгоритмов

На практике часто приходится встречаться с программами, со­держащими ошибки. Например, в самой последней операционной системе Windows специалистами обнаружено много ошибок, кото­рые время от времени выявляются на ЭВМ.

Программа **содержит ошибки,** если ее выполнение на ЭВМ при­водит к получению сбоев, отказов или неправильных результатов. Программу в таком состоянии нельзя использовать для решения практических задач.

Проявления ошибок:

**Программа**

↓

**данные → ЭВМ → { отказ | сбой | ошибка }**

**Отказ** - это ситуация, когда выполнение программы прекраща­ется вообще. Программы, содержащие такого рода ошибки считаются неработоспособными, и от их использования следует отказываться.

**Сбой -** это потеря части данных либо получение непредусмот­ренных данных. Такого рода ошибки говорят о их частичной нера­ботоспособности программ либо об их недостаточной надежности.

**Результат неправильный,** если он не соответствует требованиям, предъявляемым к работе программ. Программы, содержащие такие ошибки, считаются работоспособными, но их применение может приводить к получению ошибочных результатов.

Оценка программ:

**Задача**

**исходное требуемое**

**данные → программа → результаты**

О правильности программ **нельзя утверждать** ничего если неиз­вестны предъявляемые к ним требования. Только при наличии стро­гих, четких спецификаций можно судить о правильности работы программ.

В качестве примера рассмотрим решение квадратного уравнения:

х2 + 3⋅х + 2 = 0.

Исходные данные - коэффициенты – а = 1, b = 3, с = 2. Требу­емые результаты - пара чисел х1 и x2, являющихся корнями уравне­ния. Посмотрим, будут ли корнями уравнения пары чисел:

а) х1 = 2, x2 = 3;б) x1 = -2, x2 = -3.

Решением уравнений являются числа, подстановка которых пре­вращает уравнение в тождество. В первом случае подстановка чисел х1 = 2, х2 = 3 в уравнение дает:

22 + 3⋅2 + 2 = 12 ≠ 0 - *неправильно,*

32 +3⋅3+2 = 20 ≠ 0 - *неправильно.*

Следовательно, числа х1 = 2, х2 = 3 не являются правильными результатами.

Подстановка в уравнение чисел х1 = -2, х2 *=* -3:

(-2)2 + 3⋅(-2) +2 = 0- *правильно;*

(-3)2 + 3⋅(-3) +2 = 0- *правильно.*

Следовательно, числа х1 = -2, х2*=* -3 являются правильными результатами.

Приведем формальную постановку задачи решения квадратных уравнений.

Постановка задачи

Решение квадратного уравнения

а⋅х2 + b⋅x + с = 0.

Дано: a, b, с - коэффициенты.

Треб.: х1, х2 - корни.

Где: а⋅х12 + b⋅х1 + с *=* 0*.*

а⋅х22 *+* b⋅х2 + с *=*0.

При: а ≠ 0.

**Наличие точной постановки** задач позволяет говорить о правиль­ности не только конечных результатов, но и различных способов и методов их решения.

**Способ правильный,** если он дает правильные результаты. Способ неправильный, если он дает неправильные результаты или не дает результатов вообще.

**Метод неправильный,** если существуют допустимые данные, для которых он дает неправильные результаты либо не дает результатов вообще.

**Метод правильный,** если он дает правильные результаты для любой задачи данного класса. Использование правильных методов служит основой для составления алгоритмов и программ, не содер­жащих ошибок.

В рассматриваемом примере решения квадратных уравнений об­щим методом является вычисление корней с помощью дискрими­нанта.

Метод решения

x1 = (-b + )/(2⋅а),

x2 *=* (-b - )/(2⋅a),

где

{ D = b2 - 4⋅а⋅с.

Правильность общих методов проверяется подстановкой расчет­ных формул в исходное уравнение. Получение тождеств в результате подстановок говорит о правильности выбранных расчетных формул.

Для первого корня х1 *=* (-b + )/(2⋅a) подстановка и тождест­венные преобразования формул дадут:

а⋅х12 + b⋅х1 + с = а⋅[(-b +)/(2⋅а)]2 + b⋅ (-b +)/(2⋅a) + с =

= (-b + )2/(4⋅а) *+* b⋅ (-b +)/(2⋅a) + с = (b +) ⋅ (-b +)/(4⋅а) + с =

= (-b2 + D)/(4⋅a) + с = (-b2 + b2 - 4⋅а⋅с)/(4⋅а) + с = -4⋅а⋅с/(4⋅а) + с = 0.

Аналогичные результаты получаются и при подстановке формулы второго корня

х2 = (-b - )/(2⋅a). После выполнения анало­гичных преобразований будет получено такое же тождество. И на основании этих проверок можно сделать заключение, что рассмот­ренный метод дает правильные результаты для любык допустимых данных.

Однако саму постановку задачи необходимо дополнить условием: b2 - 4⋅а⋅с ≥ 0. При нарушении этого условия не только уравнение не имеет решений, но и метод решения также не дает результатов из-за необходимости вычисления корней от отрицательного дискриминан­та: D < 0.

В силу выбранного метода решения и принятой постановки алго­ритм решения квадратных уравнений приобретает следующий вид:

***алг «квадратное уравнение»*** Результаты вычислений

***нач***

***если а ≠ О то***при а ≠ 0

***D:******= b\*b - 4\*а\*с*** D = b2 - 4⋅а⋅с

***если D* > = *0 то*** при D *>=* 0

***х1:* = *(-b******+**)/(2\*a)*** х1 = (-b + )/(2⋅a)

***х2:* = *(-b*** *-* ***)/(2\*a)*** х2 = (-b - )/(2⋅a)

***все***

***инеc а* = *0 то*** при а = 0

***если b* ≠ *0***при b ≠ 0

***х 1: = -c/b*** xl = -c/b

***все***

***кон***

Результаты выполнения алгоритма приведены справа. Можно заметить, что результаты выполнения совпадают с описанием вы­бранного метода решения с помощью дискриминанта. Это позволяет утверждать, что алгоритм - правильный.

Алгоритм **содержит ошибки,** если можно указать допустимые ис­ходные данные, при которых либо будут получены неправильные результаты, либо результаты не будут получены вовсе. Использование алгоритмов, содержащих ошибки, приводит к созданию программ, также содержащих ошибки.

**Алгоритм** считается **правильным,** если он дает правильные резуль­таты для любых допустимых исходных данных. Правильность алго­ритмов решения прикладных задач и наличие в них ошибок можно проверять двумя основными способами.

Первый способ - проверка основных этапов построения алго­ритма:

**задача → постановка → метод → алгоритм**

Второй способ - анализ результатов выполнения алгоритмов и их сравнение с выбранными методами решения и постановкой задачи:

**задача ← постановка ← метод ← алгоритм**

Приведем пример построения алгоритма с одновременным ана­лизом его правильности.

**Задача:** Определить периметр треугольника, заданного на плос­кости координатами вершин.

XС,УС

XА,УА Xв,Ув

Постановка задачи

Определение периметра треугольника, заданного на плоскости.

Дано: А *=* (ХА, УА)

В= (ХВ, УВ) - координаты вершин треугольника

С = (XС,УС)

Треб.: Р - периметр

Метод решения

Р = LАВ +LВС+LСА

LАВ = 

LВС *=* 

LСА = 

Где: Р = L(A,B) + L(B,C) + L(C,A);

здесь L[(x,y),(u,v)] *=*  .

Приведем алгоритм, полученный из описания метода упорядоче­нием операций вычисления длин сторон треугольника с заверша­ющим вычислением периметра. Результаты выполнения алгоритма приведены справа.

***алг «периметр треугольника»***

***нач***

***LAB: =*** **

***LBC : =*** 

***LCA*** : = 

***Р := LAB + LBC + LCA***

***кон***

Результаты

**





Р = LAB + LBC + LCA

Сравнение результатов выполнения алгоритма с описанием метода решения показывает, что это одна и та же система формул, что подтверждает правильность алгоритма.

Систематические методы анализа правильности алгоритмов и программ опираются на сопоставление тех же самых описаний, которые используются при их систематическом составлении.

Анализ правильности:

**задача ← способ**

**↓ ↓**

**постановка ← методы**

**↓ ↓**

**сценарий ← алгоритмы**

**↓ ↓**

**ЭВМ → программа**

**Основные типы** алгоритмических ошибок в программах:

* ошибки в выбранных методах решения;
* ошибки в постановке решаемых задач;
* дефекты в сценариях диалога с ЭВМ;
* ошибки организации ввода данных;
* неправильная реализация методов решения.

**Исчерпывающий анализ** правильности алгоритмов и устранение из программ ошибок всех перечисленных типов возможны только при наличии соответствующих спецификаций: постановок задач, описаний методов решения и спецификаций ввода-вывода данных.

Будем считать, что **программа правильная,** если она дает правильные результаты для любых допустимых исходных данных. Такого рода программы вполне можно использовать для решения прикладных задач.

Программа считается **надежной,** если она не дает сбоев и отказов ни при каких исходных данных. Надежность - обязательное условие для всех программ, которые используются людьми для решения практических задач на ЭВМ.

В качестве иллюстрации приведем пример систематического со­ставления алгоритма и программы задачи определения суммарного веса учеников по данным из таблицы:

**фамилия рост вес**

|  |  |  |
| --- | --- | --- |
| Иванов | 185 | 85 |
| Петрова | 165 | 65 |
| Сидоров | 170 | 80 |

Рассмотрим постановку задачи и метод вычисления суммарного веса.

Постановка задачи

Определение суммарного веса.

Дано: Метод вычисления

(D1,.., DN) - данные об учениках, S0 = 0

где D = [Fam,R,V] - состав данных, Sk = Sk-1 + vk

Fam - фамилия, R - рост, V - вес. [k = (1 ... N)]

Треб.: Vsum - суммарный вес. Vsum = SN

Vsum = v1 + v2 + ... + vN

При: N > 0.

Правильность метода вычислений можно доказать по индукции. Рассмотрим результаты вычислений на 1-м, 2-м и k-м шагах. Отме­тим, что начальное значение S0 = 0.

На первом шаге при k = 1 результат вычисления

S1 = S0 +v1 = v1

На следующем втором шаге при k = 2 результат

S2 = S1 + v2 = v1 + v2.

На третьем шаге при k = 3 результат

S3= S2 + v3 = v1 + v2 + v3.

В общем случае можно предположить, что к k-му шагу результат вычисления

Sk-1=v1+...+vk-1.

Тогда результат вычислений после k-го шага (исходя из описания метода)

Sk = Sk-1 +vk = v1 + … + vk-1 *+* vk*.*

В силу принципа математической индукции утверждение верно для всех k = 1, 2,.... N. Следовательно, на последнем шаге при k = N конечный результат:

SN = v1 + ... + vN.

Что и требовалось. Следовательно, метод правильный.

Приведем сценарий диалога решения поставленной задачи на ЭВМ. Для представления данных в программе примем последова­тельность операторов data.

Сценарий Представление данных

*Данные об учениках*

*фамилия вес рост*

**dano:'данные учеников**

<Fam1> <V1> <R1> **data «Иванов», 185, 85**

… … … **data «Петрова», 165, 65**

<FamN> <VN> <RN> **data «Сидоров», 170, 80**

**data «», 0, 0**

*суммарный вес* = <Vsum>

Алгоритм обработки данных и программа, соответствующие выбранному сценарию и методу вычисления:

**Алгоритм Программа**

***алг «суммарный вес» '* суммарный вес**

***нач* cls**

***вывод («Данные об учениках»)* ? «Данные об учениках»**

***вывод («фамилия вес рост»)* ? «фамилия вес рост»**

***s := 0* s *=* 0**

***цикл* do**

***чтение famS, r, v* read fam$, r, v**

***при fam$=«» выход* if fam$=«» then exit do**

***вывод (fam$, v, r)* ? fam$; v; r**

***s := s + v* s = s + v**

***кцикл* loop**

***vsum* = *s* vsum = s**

***вывод («суммарный вec=»,vsum)* ? «суммарный вес=»; vsum**

***кон* end**

Правильность приведенного алгоритма можно увидеть из описа­ния результатов его выполнения.

**Алгоритм Результаты выполнения**

***алг «суммарный вес»* на экране и в памяти ЭВМ**

***нач***

***вывод («Данные об учениках»)* Данные об учениках**

***вывод («фамилия вес рост»)* фамилия вес рост**

***s: = 0*** s0 *=* 0

***цикл***

***чтение fam$, r, v***

***при fam$=«» выход***

***вывод (fam$, v, r)*** <famk> <vk> <rk>

***s: = s + v*** sk = sk-1 + vk

***кцикл*** [k = (1...n)]

***vsum* = *s*** vsum = sn

***вывод («суммарный вec=»,vsum)* суммарный вес= <vsum>**

***кон***

Сопоставление описания результатов выполнения с описаниями сценария и выбранного метода говорит об их полном соответствии. Следовательно, составленные алгоритм и программа правильные.

**В о п р о с ы**

1. Когда программы содержат ошибки?

2. Что такое правильный способ решения?

3. Когда способ решения неправильный?

4. Что такое правильный метод решения?

5. Когда метод решения неправильный?

6. Что такое правильный алгоритм?

7. Когда алгоритм содержит ошибки?

8. Каковы основные типы ошибок в программах?

**З а д а ч и**

1. Приведите постановку задачи, сценарий, алгоритм и программу ре­шения линейного уравнения а⋅х + b = 0, с помощью формулы х = -b/а (при а ≠ 0).

2. Приведите постановку задачи, сценарий, алгоритм и программу решения квадратного уравнения а⋅х2 + b⋅x + с = 0 с помощью формулы дискриминанта.

3. Приведите постановку задачи, сценарий, алгоритм и программу решения системы из двух уравнений с двумя неизвестными:

а⋅х + Ь⋅у = е,

с⋅х + d⋅y = f.

Примените для этой задачи вычисление корней с помощью опреде­лителей:

х = Dx/D,

y = Dy/D.

Определители D, Dx и Dy вычисляются по формулам:

D = a⋅d - b⋅c,

Dx = e⋅d - f⋅b,

Dy = a⋅f - c⋅e.

4. Приведите постановку, сценарии, алгоритм и программу решения следующих задач:

а) определение площади треугольника по длине сторон а, Ь, с по формуле Герона:

S = ,

р = (а + b + с)/2.

б) определение площади треугольника, заданного на плоскости ко­ординатами своих вершин: (х1, у1), (х2, у2), (х3, у3); для вычисления длин сторон треугольника воспользуйтесь формулой определения длин от­резков на плоскости, задаваемых координатами концов:

*l* = 

5. Приведите постановку, метод, сценарий, алгоритм и программу решения следующих задач:

а) определение времени встречи пешеходов, двигающихся навстречу друг другу;

б) определение времени, которое требуется пешеходу, чтобы догнать другого пешехода;

в) определение времени движения парохода по течению и против течения реки;

г) определение времени движения пешеходов навстречу друг другу, если один из них движется с замедлением;

д) определение времени падения тела с заданной высоты;

е) определение времени полета тела, брошенного вверх;

ж) определение расстояния, на которое улетит мяч, брошенный под углом к горизонту.

6. Дана прямоугольная матрица АNM - прямоугольная числовая таб­лица размера N × М. Приведите постановку, метод решения, сценарий, алгоритм и программу для решения следующих задач:

а) подсчет сумм элементов матрицы по столбцам,

б) подсчет сумм элементов матрицы по строкам,

в) нахождение минимального значения в каждом столбце,

г) нахождение минимального значения в каждой строке,

д) нахождение максимального значения в каждом столбце,

е) нахождение максимального значения в каждой строке,

ж) нахождение наибольшего из минимальных значений в столбцах,

з) нахождение наименьшего из максимальных значений в строках.

## 5.3. Решение прикладных задач

**Решение задач на ЭВМ** является одним из основных источников для создания алгоритмов и программ. Экономические задачи и про­блемы обработки данных - один из важнейших классов приклад­ных задач, решаемых на ЭВМ.

**Применение компьютеров** для решения экономических задач су­щественно упрощает работу по подготовке и обработке данных. Одной из причин в использовании ЭВМ для решения этих задач - снижение трудоемкости и уменьшение числа ошибок при обработке данных.

Для решения многих экономических задач на ЭВМ используются электронные таблицы и специальные пакеты программ. Однако решение **любых новых прикладных задач** на ЭВМ предполагает необ­ходимость создания новых алгоритмов и программ на основе опре­деленных математических методов решения и обработки данных.

Особое значение **правильность алгоритмов** имеет для экономичес­ких задач, поскольку ошибки в их решении могут дорого стоить. Неправильные экономические расчеты могут нанести материальный ущерб или даже привести к банкротству целую организацию.

Для **предотвращения ошибок** можно использовать систематические методы конструирования алгоритмов и программ с одновременным анализом их правильности. Последовательное применение этих методов обеспечивает составление прикладных алгоритмов и про­грамм с гарантиями их правильности.

Общий принцип **систематического подхода** к составлению алго­ритмов и программ заключается в последовательной разработке спецификаций: постановок задач, способов и методов их решения, а также сценариев работы в процессе решения задач.

**Составление программ**

**задача → способы**

**↓ ↓**

**постановка → методы**

**↓ ↓**

**сценарий → алгоритмы**

**↓ ↓**

**ЭВМ ← программы**

**Систематический анализ** правильности алгоритмов и программ сводится к сопоставлению этих спецификаций друг с другом: про­грамм - с алгоритмами, алгоритмов - со сценариями и описаниями методов, а методы решения - с постановками задач.

**Анализ правильности**

**задача ← способ**

**↑ ↑**

**постановка ← методы**

**↑ ↑**

**сценарий ← алгоритмы**

**↑ ↑**

**ЭВМ → программы**

Приведем примеры систематической разработки алгоритмов и программ решения экономических задач на ЭВМ с обоснованием их правильности. Главной особенностью этих задач является то, что все они относятся к задачам обработки данных.

Первый пример экономической задачи - определение средней зарплаты в организации. Допустим, что данные о зарплате представ­лены таблицей:

**фамилия должность зарплата**

|  |  |  |
| --- | --- | --- |
| Иванов | директор | 300000 |
| Петров | менеджер | 240000 |
| Сидорова | секретарь | 120000 |

Приведем постановку задачи и описание метода вычисления сред­ней зарплаты.

Постановка задачи Метод расчета

Определение средней зарплаты.

Дано:

(D1, ..., DN) - данные о сотрудниках,

где D = [Fam, Т, Z] - состав данных,

Fam - фамилия, D1*-* должность, S0 = 0

Z - зарплата. Sk = Sk-1\*(k-l )/k + Zk/k

Треб: Zcpeдн - средняя зарплата. [k=(l...N)]

Где: Zcpeдн = (Z1 +Z2 + ... + ZN)/N. Zcpeдн = SN

При: N > 0.

Прежде всего убедимся, что выбранный метод вычисления пра­вилен. Для этого воспользуемся индукцией. Рассмотрим результаты вычислений на первых трех шагах.

При k = 1 результат

S1=S0(1 - 1)/1 +Z1/1 =Z1/1.

При k = 2 результат

S2 = S1(2 - 1)/2 + Z2/2 = Z1/2 + Z2/2.

При k = 3 результат

S3 = S2(3 - 1)/3 + Z3/3 *=* (Z1 + Z2)/3 + Z3/3.

По этим трем результатам можно утверждать, что в общем случае результатом k-го шага вычислений будет

Sk = (Z1 + ... + Zk-1)/k.

Справедливость этого утверждения можно доказать по индукции. Допустим, что оно справедливо для (k-l)-ro шага:

Sk-1 = (Z1 + ... + Zk-1)/(k-l).

Тогда из описания метода вычислений очередное k-e значение будет равно

Sk *=* Sk-1(k-l)/k + Zk/k *=*

*=* (Z1 + ... + Zk-1)/(k-l)⋅(k-l)/k + Zk/k = (Z1 + ... + Zk-1)/k + Zk/k.

Что и требовалось показать. Следовательно, в силу математичес­кой индукции это утверждение справедливо для всех k = 1, 2,..., N. В частности, для последнего шага вычислений при k = N конечным результатом будет

SN = (Z1 + ... + ZN-1)/N + ZN/N = (Z1 + ... + ZN)/N.

Таким образом, выбранный метод дает правильный результат для любой последовательности величин Z1, Z2, ..., ZN.

Для конструирования алгоритма и программы решения задачи на ЭВМ примем следующий сценарий, а для представления данных воспользуемся операторами **data.**

Сценарий Представление данных

***список сотрудников:* dan: 'данные сотрудников**

***{<фам> <должн> <з/плата>}\** data «Иванов»,«директор», 300000**

***{...................}*  data «Петров»,«менеджер», 240000**

***средняя з/плата= <Zcpeд>* data «Сидорова»,«секретарь», 120000**

**data «», «», 0**

При выбранных сценарии, методе расчета и представлении данных систематическое конструирование приводит к следующим алгоритму и программе.

**Алгоритм Программа**

***алг «средняя зарплата»* ' средняя зарплата**

***нач* cls**

***вывод («список сотрудников:»)* ? «список сотрудников:»**

***s* := *0: k* := *0* s = 0: k = 0**

***цикл* do**

***чтение (fam$, dl$, zpl)* read fam$, dl$, zpl**

***при fam$ =*** *«»* ***выход* if fam$ = «» then exit do**

***вывод (fam$, dl$, z)* ? fam$; dl$; z**

***k* := *k + 1* k = k + 1**

***s* := *s\*(k - 1)/k* + *z/k*  s *=* s\*(k - 1)/k + z/k**

***кцикл* loop**

***zsr* = *s* zsr = s**

***вывод («средняя 3/nлama=»,zsr)* ? «средняя з/плата=»; zsr**

***кон* end**

Для полного обоснования отсутствия ошибок в приведенном алгоритме и программе приведем описание результатов их выполне­ния на ЭВМ.

**Алгоритм Результаты выполнения**

***алг «средняя зарплата»***

***нач***

***вывод («список сотрудников:»)* список сотрудников:**

**s := *0: k*** **:= *0***S0 = 0 [ k = 0 ]

***цикл***

***чтение (fam$, dl$, z)***

***при fam$ = «» выход***

***вывод (fam$, dl$, z)*** <famk> <dlk> <zk> }\*

***k:=k***+ ***1*** [ k= (1...N) ]

***s := s\*(k - 1)/k******+ z/k*** sk = sk - 1⋅(k - 1)/k + zk/k

***кцикл***

***zsr*** = *s* zsr *=* sN

***вывод («средняя з/nлama=»,zsr)* средняя з/плата= <zsr>**

***кон***

Сравнение результатов выполнения программы с описанием метода вычисления и выбранного сценария подтверждает их соот­ветствие друг другу и как следствие правильности выбранного метода вычислений - правильность составленных алгоритма и программы расчета средней зарплаты.

В качестве второго примера рассмотрим решение типичной задачи подсчета суммарной стоимости товаров с выделением товаров наибольшей стоимости. Допустим, что исходные данные представ­лены следующей таблицей:

**товар цена кол-во**

|  |  |  |
| --- | --- | --- |
| яблоки | 8000 | 3 |
| бананы | 4000 | 2 |
| арбузы | 1000 | 20 |

Приведем постановку задачи и описание способа ее решения.

Постановка задачи Способ решения

Определение суммарной

и максимальной стоимости товаров.

Дано:

(D1, ..., DN) - данные о товарах,

где D = [Tov, C, M] - состав данных, s0 = 0

Tov - товар, С - цена товара, от k = 1 до N цикл

М - количество товара, sk = sk-1 + СkМk

Треб: если k = 1 то

Sum - суммарная стоимость товаров, mах1 = С11М11

TovMax - товар максимальной инеc СkМk > mахk-1 то

стоимости.

Где: mахk = СkМk

Sum = C1M1 + С2М2 + ... + СNМN, все

TovMax: C⋅M = Мах(С1М1, ... ,СNМN). кцикл

При: N > 0.

Прежде чем приступить к составлению алгоритмов и программ, убедимся в правильности выбранного способа решения. Для этого проверим результаты на первых шагах, в середине и в конце вычис­лений. На первом шаге при k = 1 результат

s1 = s0 + С1М1 = С1M1,

max1 = С1М1.

На втором шаге вычислений будут получены следующие значе­ния:

s2 = s1 + С2М2 = C1M1 + С2М2,

max2 = С2М2, при С2М2 > max1 = Мах(mах1, С2М2),

max1, при С2М2 ≤ max1 = Мах(mах1, С2М2).

На третьем и последующих шагах в общем случае будут получать­ся результаты:

sk = sk-1 + CkMk = C1M1 + … + CkMk,

maxk = Max(maxk-1, СkМk) = Мах(С1М1, ..., СkМk).

Для доказательства этих утверждений необходимо предположить, что они выполняются для случая k-1:

sk-1 =C1M1 +...+ Ck-1Mk-1,

maxk-1 = Max (C1M1, …,Ck-1Mk-1),

и подставить эти выражения в соотношения для sk и mахk:

sk = sk-1 + CkMk = C1M1 + … Ck-1Mk-1 + CkMk,

maxk = Max(maxk-1, СkМk) = Мах(С1М1, ..., СkМk).

В силу математической индукции эти утверждения верны для всех k *=* 1, 2, ..., N. Поэтому на последнем шаге вычислений при k = N будут получены окончательные результаты:

sN = sN-1 + CNMN = C1M1 + … + CNMN,

maxN = Max(maxN-1, СNМN) = Max(C1M1, ... , СNМN).

**Что и требовалось** в постановке задачи. Следовательно, выбран­ный способ решения поставленной задачи правилен и на его основе можно приступать к составлению соответствующих алгоритма и программы.

Для систематичности разработки примем следующий сценарий диалога и представление исходных данных в операторах **data.**

Сценарий Представление данных

*список товаров*

*товар цена кол-во*

*<тов1> <с1> <т1> \** **dan: 'сведения о товарах**

… .... ... **data яблоки, 8000, 3**

*сумма = <Sum>* **data бананы, 4000, 2**

*Максимум* **data арбузы, 1000, 20**

*<товар> <стоим>* **data «», 0, 0**

Приведем алгоритм и программу решения поставленной задачи в соответствии с выбранным сценарием и представлением данных.

***Алгоритм* Программа**

***алг «сумма и максимум» '* сумма и максимум**

***нач* сls**

***вывод («список товаров»)* ? «список товаров»**

***вывод («товар цена кол-во»)* ? «товар цена кол-во»**

***s := 0; k = 0* s = 0: k = 0**

***цикл* do**

***чтение (тов, с, т)* read tv$, с, m**

***при тов* = *«» выход* if tv$ = «» then exit do**

***k := k + 1* k = k + 1**

***вывод (тов, с, т)* ? fv$; с; m**

***s :=s + cm* s= s *+* c(m**

***если k = 1 то* if k = 1 then**

***max* := *c⋅m* max = c*⋅*m**

***ToвMax* := *тов* ТМ$ = tv$**

***инес c(m > max то* elseif c(m > max then**

***max* := *c⋅m* max = c*⋅*m**

***ToвMax := тов* TM = tv$**

***кесли* end if**

***кцикл* loop**

***вывод («cyммa=»,s)* ? «cyммa=»,s**

***вывод («Максимум»)* ? «Максимум»**

***вывод (ToвMax, max)* ? TM$, max**

***кон* end**

Сравнение результатов выполнения представленных алгоритма и программы с описанием выбранного способа решения показывает их полное соответствие друг другу.

**Алгоритм Результаты выполнения**

***алг «сумма и максимум»***

***нач***

***вывод («список товаров»)* список товаров**

***вывод («товар цена кол-во»)* товар цена кол-во**

***s* :=*0*; *k = 0*** s0 =0 [k = 0]

***цикл***

***чтение (тов, с, т)***

***при тов = «» выход***

***k:=k+1*** [k= 1,2,...,N]

***вывод (тов, с, т)*** { <тов> <с> <m> }\*

***s := s + с⋅т***sk *=* sk-1 + ck⋅mk

***если k*** *=****1 то*** при k = 1

***тах* := *c⋅m*** max1 = c1⋅m1,

***ТовМах* := *тов*** ToвMaх1 *=* тов1

***uнес c⋅m > тах то*** при сk⋅mk > mах

***тах* := *с⋅т*** mахk = сk⋅mk

***ТовМах* := *тов*** ТовМахk = товk

***кесли***

***кцикл***

***вывод («сумма=», s)***  **cуммa = <sN>**

***вывод («Максимум»)* Максимум**

***вывод (ТовМах, тах)* <ToвMaxN> <maxN>**

***кон***

Из расмотренных примеров следует, что правильность алгоритмов и программ зависит прежде всего от правильности выбранных методов решения. Составление соответствующих им алгоритмов и программ сводится к решению технических проблем.

Можно утверждать, что правильные алгоритмы и программы - это корректная реализация **правильных** **методов** решения. Ошибки в выбранных методах решения носят не алгоритмический, а принципиальный характер и их следует искать не с помощью отладки программ на ЭВМ, а исследованием самих методов.

Рассмотрим самую популярную экономическую задачу - **расчет семейного бюджета** в целях анализа достатка семьи. Напомним, что достаток семьи - это остаток от разности доходов и расходов:

достаток = доходы - расходы.

Допустим, что данные о семейном бюджете представлены двумя таблицами: - таблицей доходов и таблицей расходов:

**Доходы Расходы**

|  |  |  |  |
| --- | --- | --- | --- |
| папа | 3000 | питание | 200 |
| мама | 1200 | одежда | 120 |
| брат | 2000 | транспорт | 60 |
| я | 600 | отдых | 30 |
|  |  | разное | 50 |

Приведем точную постановку задачи и опишем метод ее реше­ния.

Постановка задачи Метод решения

Определение достатка семьи.

Дано: S *=* Sd *-* Sr

D = (дох1, ..., дох N) - доходы, Sd = сN

R = (расх1, ..., расхМ) - расходы, сk = сk-1 + dk

где дох *=* (имя, d), [k = (1...N)]

расх = (стат, r). с0 = 0

Треб.: S - достаток семьи. Sr = bM

Где: bi *=* bi-1 + ri

S = Sum (d1, …, dN) - Sum (r1, .... rM). [i = (1 ... M)]

При: N, M > 0. b0 = 0

Для решения задачи на ЭВМ в качестве представления данных примем два списка операторов **data,** а для организации вывода ре­зультирующих данных - следующий сценарий.

Сценарий Представление данных

*Подсчет достатка* *'***doch: ' доходы**

*Доходы семьи:* **data «папа», 300000**

*<имяk>* <dk> \* **data «мама», 120000**

**... ... data «брат», 200000**

*Доходов* = *<Sd>* **data** **«», 0**

*Расходы семьи:*

***<****статk> <rk****> \** rash: ' расходы**

**... ... data «питание», 200000**

*Расходов = <Sd>* **data «одежда», 120000**

*Достаток* *= <S>* **data «транспорт», 60000**

**data «», 0**

Приведем соответствующие этому сценарию и выбранному методу представления данных алгоритмы и программу на Бейсике:

***алг «достаток семьи»* 'достаток семьи**

***нач* cls**

***вывод («Подсчет достатка»)* ? «Подсчет достатка»**

***вывод («Доходы семьи:»)* ? «Доходы семьи:»**

***подсчет\_доходов* gosub dchs 'доходы**

***вывод («Доходов=», Sd)* ? «Доходов=», Sd**

***вывод («Расходы семьи:»)* ? «Расходы семьи:»**

***подсчет\_расходов* gosub rashs 'расходы**

***вывод («Расходов =», Sr)* ? «Расходов=», Sr**

***S* := *Sd - Sr* S = Sd - Sr**

***вывод («Достаток=», S)* ? «Достаток=», S**

***кон* end**

***алг «подсчет доходов»* dchs: 'подсчет доходов»**

***нач* '**

***загрузка\_доходов* restore doch 'доходы**

***Sd := 0* Sd = 0**

***цикл* do**

***чтение (имя, d)* read namS, d**

***при имя* = *«» вых* if nam$ = «» then exit do**

***вывод (имя, d)* ? nam$, d**

***Sd = Sd + d* Sd = Sd + d**

***кцикл* loop**

***кон* return**

***алг «подсчет расходов»* rashs ' подсчет расходов**

***нач* '**

***загрузка\_расходов* restore rach 'расходы**

***Sr := 0*****Sr = 0**

***цикл* do**

***чтение (стат, r)* read stat$, r**

***при стат* = *«» вых* if st$ = «» then exit do**

***вывод (стат, r)* ? st$, r**

***Sr = Sr + r* Sr = Sr + r**

***кцикл* loop**

***кон* return**

Правильность составленного комплекса алгоритмов и программы расчета достатка семьи можно проверить по описанию результатов их выполнения:

«достаток семьи» «подсчет доходов» «подсчет расходов»

Подсчет достатка

Доходы семьи: Sd0 = 0 [k = 0] Sr0 = 0 [i = 0]

<подсчет\_доходов>

Доходов = <Sd>

Расходы семьи: [k =(1...N)] [i =(1...M)]

<подсчет\_расходов> <имяk> <dk> <стат1> <r1>

Расходов = < Sr> Sdk = Sd/k-l/+dk Sri == Sri-1 + ri

{ S = Sd - Sr

Достаток = <S>

Для обоснования правильности всего комплекса алгоритмов и программы в целом необходимо показать правильность каждого из вспомогательных алгоритмов: «подсчет доходов» и «подсчет расходов».

Для первого алгоритма для первых шагов вычисления получаем:

Sd0 = 0,

Sd1 = Sd0 + d1 = d1,

Sd2 = Sd1 + d2 = d1 + d2.

Для последующих шагов можно заключить, что

Sdk = Sdk-1 + dk = d1 + d2 *+ ... +* dk-1 + dk.

Это доказывается с помощью математической индукции. В силу этого утверждения окончательным результатом вычислений станет сумма доходов

SdN = d1 + d2 + ... + dN-1 + dN.

Следовательно, алгоритм подсчета доходов - правильный.

Для второго алгоритма подсчета расходов получаются аналогич­ные оценки:

Sr0 = 0,

Sr1 = Sr0 + r1 = r1,

Sr2 = Sr1 + r2 = r1 + r2

и для последующих шагов вычислений:

Sri = Sri-1 + ri = r1 + r2 +... + ri-1+ ri.

Это доказывается также с помощью математической индукции. На основании этого утверждения можно сделать заключение о ко­нечном результате выполнения алгоритма:

SrM = r1 + r2 + ... + rM-1+ rM.

Следовательно, алгоритм подсчет расходов правильный. Но в основном алгоритме содержится единственная расчетная формула

S *=* Sd - Sr.

В силу доказанных утверждений о результатах выполнения алго­ритмов «подсчета доходов» и «подсчета расходов» конечным резуль­татом вычислений станет величина

S = Sd - Sr = (d1 + d2 + ... + dN) - (r1 + r2 + ... + rM).

Что и требовалось доказать. Следовательно, весь комплекс алго­ритмов и программа в целом правильны.

**В о п р о с ы**

1. К чему приводят ошибки в экономических программах?

2. Кто отвечает за ошибки в экономических программах?

3. Что дают постановки задач?

4. Зачем нужны описания методов?

5. Как проверяется правильность методов?

6. Зачем нужны описания результатов?

**З а д а ч и**

1. В магазине имеются товары различных наименований. В течение дня каждый из М покупателей (М - заданное число) сообщил о своем намерении приобрести определенное количество товара одного из на­именований. Требуется определить суммарный спрос на товары каждого наименования, расположив товары в порядке убывания дневного спроса на них.

2. Каждый из N магазинов в течение месяца работал D дней (N и D - заданные числа 1, 2, .... N). Известна прибыль каждого магазина в каждый день его работы. Необходимо напечатать упорядоченный по месячным доходам список названий магазинов, имеющих прибыль в пересчете на один день работы выше средней дневной прибыли по всем магазинам.

3. Каждое из N предприятий города выпускает М одинаковых на­именований продукции (N и М, наименования продукции и названия предприятий известны). Заданы объем выпуска и стоимость единицы продукции каждого вида для каждого из предприятий. Необходимо для каждого вида продукции определить предприятия, выпускающие наибольший объем этой продукции.

4. Из разных городов выбрали N семей (N - заданное число). Каждая семья характеризуется числом членов и доходом каждого изних.Для каждого города сформировать перечень семей с минимальным доходом в пересчете на отдельного члена семьи, указав порядковые номера семей из общего списка.

5. Ассортимент N магазинов состоит из М товаров (N, М и названия товаров заданы). Каждый товар характеризуется наличием или отсутст­вием его в магазине, а также наличием или отсутствием на него спроса покупателей. Требуется перечислить названия ходовых (есть спрос и товар имеется хотя бы в одном магазине), неходовых (спрос отсутствует, а товар имеется хотя бы в одном магазине) и дефицитных (спрос есть, а товара нет ни в одном из магазинов) товаров.

## 5.4. Элементы доказательного программирования

**Доказательное программирование** - это составление программ с доказательством их правильности. Сложность составления и доказа­тельства правильности алгоритмов и программ состоит в следующем.

Для заключений о **наличии ошибок** в алгоритме или в программе достаточно указать тест, при котором произойдет сбой, отказ или будут получены неправильные результаты. Поиск и исправление ошибок в программах обычно проводится на ЭВМ.

Для утверждений о **правильности программ** необходимо показать, что правильные результаты будут получаться для всех допустимых данных. Такие утверждения могут быть доказаны только путем исчерпывающего анализа результатов выполнения программ при любых допустимых данных.

Существуют **два подхода к проверке** программ - прагматический и доказательный. При прагматическом подходе проверка программ выполняется на ЭВМ путем тестирования.

**Тестирование** - это проверка программ на ЭВМ с помощью не­которого набора тестов. Ясно, что тестирование не дает гарантий правильности выполнения программ на всех допустимых данных. Следовательно, тестирование в общем случае не может дать и не дает полных гарантий отсутствия ошибок в программах.

Напомним, что **отладка программ** - это процесс поиска и исправ­ления ошибок в программах на ЭВМ. Однако поскольку поиск оши­бок при отладке программ проводится с помощью тестов, то полных гарантий нахождения и исправления всех ошибок в программах отладка не дает и в принципе дать не может.

По этой же причине **не ясно,** когда процесс отладки программ - процесс поиска и исправления ошибок на ЭВМ - может считаться завершенным. А выявлены или нет все ошибки в программе при ее отладке не может сказать никто.

Таким образом, прагматический подход чреват созданием про­грамм, содержащих ошибки даже после «завершения» отладки, что и наблюдается практически во всех больших программах для ЭВМ.

Рассмотрим в качестве иллюстрации принципов тестирования алгоритм и программу вычисления максимума из трех чисел: а, b, с.

***алг «максимум трех чисел»* 'максимум трех чисел**

***нач* cls**

***ввод (а, b, с)* input a, b, с**

***если а > b то* if а > b then**

***тах := a* max = a**

***инеc b > с то* elseif b > с then**

***тах := b* mах = b**

***инеc с > а то* elseif с > a then**

***тах*:= с mах = с**

***кесли* end if**

***вывод («тах=»,тах)* ? «mах=»; mах**

***кон* end**

Запуск этой программы на ЭВМ можно проверить на следующих данных:

**Tecт1 Тест2 Тест3**

? 1 1 2 ? 1 2 3 ? 3 2 1

max = 2 max *=* 3 max *=* 3

Все три результата правильные. Отладку программы после запуска этих примеров можно было бы считать завершенной. Однако есть контрпример:

**Контрпример1**

? 2 1 3

max = 2

Но этот результат - неправильный. Следовательно, алгоритм и программа содержат ошибки. Но сколько этих ошибок - одна, две, а может быть больше?

При **доказательном подходе** разработка алгоритмов и программ предполагает составление спецификаций и доказательство их правильности по отношению к этим спецификациям. Процесс разработки программ считается завершенным после проверки их на ЭВМ и предоставлении доказательств отсутствия ошибок.

**Доказательства правильности** алгоритмов и программ, равно как и любые другие доказательства, строятся на основе суждений и рас­суждений. В данном случае суждения и рассуждения касаются результатов выполнения алгоритмов и программ с теми или иными данными.

Конструктивно, **доказательства** правильности алгоритмов и про­грамм строятся на суждениях и утверждениях о результатах выпол­нения каждого из составляющих их действий и операций в соответ­ствии с порядком их выполнения.

В качестве примера проведем анализ результатов алгоритма, со­стоящего из трех присваиваний.

***алг******«у* = *х5»*** Результаты Утверждения

***нач***

***v* := *х⋅х*** v1 = х⋅х ⇒ v1 = x2

***v := v⋅v*** v2 = v1⋅v1 ⇒ v2 = x4

***у := v⋅x*** у = v2⋅x ⇒ у = х5

***кон***

Справа от алгоритма приведены результаты выполнения присва­иваний. Результатом первого присваивания v := х⋅х будет значение v1 = х⋅х переменной v. Результат следующего присваивания v := v⋅v - второе значение переменной v, равное v2 = v1⋅v1 . Результатом треть­его присваивания у := v⋅x будет значение у = v2⋅x .

На основе приведенных рассуждений, можно сделать три утверж­дения о промежуточных и конечных результатах вычислений:

Результаты Утверждения

{ v1 = х⋅х ⇒ v1 = х2

{ v2 = v1⋅v1 ⇒ v2 = x4

{ у = v2⋅x ⇒ у = х5

Таким образом можно высказать окончательное

**Утверждение.** Конечным результатом выполнения будет

**у = х5** для любых значений х.

**Доказательство.** Исходя из описания результатов выполнения присваиваний значение у будет равно

у = v2⋅x = (v1⋅v,)⋅x = ((х⋅х).(х⋅х))) ⋅х = x5.

**Что** и **требовалось** доказать.

Техника **анализа** и **доказательства** правильности алгоритмов и программ во многом совпадает с техникой доказательства любых других утверждений и состоит в применении следующих четырех приемов:

* разбор случаев;
* подбор контрпримеров;
* выделение лемм;
* индуктивный вывод.

**Разбор случаев** применяется для анализа результатов выполнения конструкций альтернативного выбора. В качестве примера проведем анализ приведенного выше алгоритма «выбора» максимума трех чисел, содержащего выбор альтернатив.

***алг «у* = *тах(а, b,с)»*** Результаты

***нач***

***если а > b то*** при а > b

***у* := *а*** у = а

***инес b > с то*** при b > с

***у* := *b*** у = b

***инес с > а то*** при с > а

***у* := *с*** у = с

***кесли***при не (b > с)

***кон***

Справа от алгоритма приведены результаты вычислений с указа­нием условий, при которых они получаются. На основании этих фактов можно заключить, что конечные результаты вычисления имеют три варианта:

а, при а > b,

у = b, при b > с и b ≥ а,

с, при с > а и с ≥ b.

В то же время значение максимума должно быть равно:

а, при а ≥ b и а ≥ с,

mах = b, при b ≥ с и b ≥ а,

с, при с ≥ а и с ≥ b.

Во всех трех случаях видны различия в условиях получения и определения максимальных значений. Покажем, что эти различия существенны и утверждение о том, что алгоритм дает правильные результаты для всех данных, неверно.

Для **опровержения общего утверждения** достаточно указать хотя бы один контрпример. В данном случае утверждение о правильности алгоритма гласило бы: для любых значений переменных а, b, с конечным было бы значение mах (а, b, с).

**Контрпримером** в данном случае будут значения: а = 2, b = 1, с = 3. Для этих данных по определению mах = 3, а по результатам выполнения алгоритма у = 2. Следовательно, в алгоритме содержится ошибка.

Однако оказывается, что это **не единственная ошибка.** Более тон­кие ошибки вскрывает второй контрпример: а = 1, b = 1, c = 1. Для этих данных в алгоритме вовсе не определен результат вычислений у = ? и конечный результат выполнения программы будет непред­сказуем!?

Правильное решение этой задачи можно получить применением систематических методов, составив постановку и описание метода решения.

Постановка задачи Метод решения

Вычисление mах (а, b, с).

Дано: а, b, с - три числа, mх = mах(mах(а,b),с)

Треб.: mх - максимум, mах(х,у) = х, при х ≥ у

Где: mх = mах (а, b, с). у, при у ≥ х

Данный метод решения непосредственно состоит из формул определения максимумов из трех и двух чисел. Реализация этого метода в форме алгоритма может быть такой:

***алг «тх* = *тах(а,b,с)»*** Результаты

***нач***

***если а*** ≥  ***b то*** при а ≥ b

***тх******:****=* ***а*** mx = a

***иначе*** при b > а

***mх******:****=* ***b*** mх = b

***кесли { mх = mах(а,b) }*** при с < mх

***если с ≥ mх то*** при с ≥ mх

***mх := с*** mх' = с

***кесли***

***кон***

**Доказательство** правильности алгоритмов можно проводить двумя способами. Первый способ - анализ правильности при по­строении алгоритмов. Второй способ - анализ правильности после построения алгоритмов.

**Первый способ** - показать, что алгоритм является корректной реализацией метода решения, и доказать, что метод - правильный. Для рассмотренного алгоритма это доказательство изложено выше.

Привлечение для создания алгоритмов известных методов реше­ния, для которых доказана их правильность, позволяет существенно упростить обоснование правильности программ. При этом центр тяжести проблем смещается к созданию и обоснованию гарантиро­ванно правильных методов решения задач.

**Второй способ** - исчерпывающий анализ результатов выпол­нения алгоритма на соответствие постановке решаемых задач для любых допустимых данных. Это - доказательство путем исчерпыва­ющего анализа результатов выполнения алгоритмов и программ.

Результаты выполнения рассматриваемого алгоритма вычисления максимума трех чисел приведены справа от него. Анализ результатов алгоритмов, содержащих конструкцию выбора, требует разбора случаев. Отметим, что все эти случаи были уже указаны ранее при разборе ошибочной версии алгоритма.

Для обоснования правильности алгоритма докажем вспомогатель­ное утверждение о результатах выполнения конструкции альтерна­тивного выбора

**Лемма:** Конечными результатами выполнения алгоритма

**Алгоритм Результаты**

***если а* > *b то*** при а ≥ b

***тх******:= а***mx = a

***иначе***при b > a

***тх* := *b*** mx = b

***кесли***

является значение mx = max(а, b) для любых значений а и b.

**Доказательство.** Результатом вычислений здесь будут значения

а при а ≥ b

mx =

b при а < b

что совпадает с определением max (а, b).

С помощью этой леммы легко доказать правильность алгоритма в целом.

**{ mх = max (а, b)** } Результаты

***если* с** ≥ **mx *то*** при с ≥ mx

**mx** **:= с** mx' = с

***кесли***mx' = mx

при с < mx

**Утверждение.** Конечным результатом выполнения алгоритма вы­числения максимума будет значение mx' = max (а, b, с) для любых значений а, b и с.

**Доказательство.** В силу предположения предшествующее значе­ние переменной mx = max(a,b). Отсюда получаем, что

с, при с ≥ mx

mx′ = = max(a,b,c).

mx, при с < mx

**Что и требовалось** доказать.

**Доказательство лемм** - основной прием доказательства правиль­ности сложных алгоритмов и программ. Напомним, что лемма — это вспомогательное утверждение, предполагающее отдельное доказа­тельство.

Одним из важнейших применений аппарата лемм является анализ результатов выполнения и доказательство правильности алгоритмов с циклами. Используемые для анализа циклов леммы называются **индуктивными утверждениями.** Эти леммы выражают утверждения о промежуточных результатах выполнения циклов.

В качестве примера использования индуктивных рассуждений рассмотрим алгоритм вычисления среднего арифметического после­довательности чисел. В приводимом алгоритме предполагается, что последовательность чисел размещена в массиве X[1:N].

***алг «среднее значение»***

***массив X[1:N]***

***нач*** Результаты:

***от k = 1 до N цикл***

**S := S \* (k-l)/k + X[k]/k** Sk = Sk-1\*(k-l)/k + X[k]/k

***кцикл*** [k = (1...N)]

***Xcp := S*** Xcp = S

***кон***

Этот алгоритм обычно считается ошибочным (?!). «Ошибкой» в этом алгоритме считается отсутствие присваивания S := 0 перед началом цикла.

Разберем результаты выполнения алгоритма на первых трех ша­гах:

S1 = S0⋅(l - 1)/1 + Х[1]/1 = S0⋅0/1 + Х[1]/1 = Х[1]/1;

S2 = S1⋅(2 - 1)/2 + Х[2]/2 = S1⋅1/2 + Х[2]/2 = Х[1]/2 + Х[2]/2;

S3 = S2⋅(3 *-* 1)/3 + Х[3]/3 = S2⋅2/3 + Х[3]/3 = Х[1]/3 + Х[2]/3 + Х[3]/3.

Можно утверждать, что на первых трех шагах результатом является среднее арифметическое обрабатываемых чисел. На основе этих примеров можно сделать **индуктивное утверждение** - «на каждом очередном k-м шаге выполнения цикла результатом будет среднее арифметическое»

Sk = Sk-1⋅(k-l)/k + X[k]/k = X[l]/k + X[2]/k + ... + X[k]/k.

Доказательство этого утверждения проводится с помощью мате­матической индукции. На первом шаге при k = 1 оно уже доказано. Допустим, что оно справедливо на (k -1)-м шаге

Sk-1 = X[l]/(k-l) + X[2]/(k-l) + ... + X[k-l]/(k-l).

Подставим его в описание результатов цикла на k-м шаге

Sk= Sk-1⋅(k-l)/k +X[k]/k.

Тогда результат выполнения цикла на k-м шаге оказывается рав­ным

Sk = X[l]/k + X[2]/k + ... + X[k-l]/k + X[k]/k,

т. е. среднему арифметическому первых k чисел.

Таким образом, индуктивное утверждение доказано. В силу мате­матической индукции это утверждение верно для всех k = l, 2, ..., N. Следовательно, на последнем шаге конечным результатом выполнения цикла станет значение

SN = SN-1⋅(N-1) + X[N]/N = X[1]/N + ... + X[N]/N.

Исходя из этого утверждения конечным результатом выполнения алгоритма в целом будет среднее арифметическое значение

Xcp = SN = X[1]/N + ... + X[N]/N.

Следовательно, приведенный алгоритм, несмотря на содержа­щуюся в нем «ошибку», является правильным. В целом анализ правильности алгоритмов с циклами во многом построен на исполь­зовании индукции.

**Индукция** - это вывод общих суждений из частных примеров. При анализе циклов она используется для подбора индуктивных утверждений о промежуточных результатах выполнения циклов. Однако для доказательства правильности индуктивных утверждений о результатах выполнения циклов используется полная математи­ческая индукция.

**Математическая индукция** - это принцип доказательства после­довательностей утверждений Р(1), Р(2), Р(3), ..., P(N), .... когда известно, что верны первые утверждения для n = 1, 2, 3 и из истин­ности (n - 1)-го утверждения следует истинность n-го утверждения:

**Принцип математической индукции:** если первое утверждение Р(1) **истинно** и из утверждения Р(n - 1) **следует** утверждение Р(n), то истинны **все** утверждения Р(1), Р(2), Р(3), ..., Р(n), ... .

Приведем примеры индуктивного анализа циклов для алгоритма нахождения минимального значения в последовательности чисел, который в этот раз действительно будет ошибочным.

***алг «нахождение минимума»***

***массив x[1:N]***

***нач*** Результаты:

***от k = 1 до N цикл***

***если x[k] < min то***

***тп := x[k]*** mnk = { x[k], при x[k] < mnk-1,

***все*** { mnk-1, в ином случае

***кцикл*** [ k = (1 ... N)]

***Min* := *тп*** Min = mnN

***кон***

**Утверждение.** Приведенный алгоритм определения минимального значения последовательности чисел неправильный.

**Доказательство.** Для демонстрации ошибок необходимо привести контрпример. Для построения контрпримера разберем результаты выполнения на первом шаге цикла.

Результаты выполнения первого шага цикла при k = 1:

х[1] при х[1] < mn0

mn1 *=* = min (х[1], mn0).

mn0 при х[1] ≤ mn0

Следовательно, результатом будет

mn1 = min (x[l], mn0)

Однако поскольку начальное значение mn0 неизвестно, то не­определено значение результата выполнения первого шага цикла. Аналогичное утверждение можно сделать о втором и всех последу­ющих шагах выполнения цикла:

mnk = min (x[k], Min(x[k-l], ..., х[1], mn0) = Min (x[k], x[k-1], ..., х[1], mn0).

В силу математической индукции это утверждение справедливо при k = N:

mnN = Min (x[N], x[N - 1], ..., x[2], х[1], mn0),

Таким образом на основании этого утверждения можно сделать заключение о конечном результате выполнения алгоритма в целом:

Min *=* mnN = Min (x[N], x[N - 1], ..., x[2], х[1], mn0).

Из этой формулы видно, что конечный результат равно как и результат первого присваивания зависит от начального значения mn0 переменной mn. Однако эта величина не имеет определенного значения, соответственнно неопределен и конечный результат выполнения алгоритма в целом, что и является **ошибкой.**

В самом деле, если значение mn0 окажется меньше любого из значений последовательности х[1], .... x[N], то конечный результат вычислений будет неправильным. В частности, при реализации алгоритма на Бейсике неправильный результат будет получен, если последовательность будет состоять только из положительных чисел. Например, для последовательности чисел: 1, 2, 3, ..., N.

Приведем правильную версию алгоритма с доказательством отсутствия ошибок, используя технику индуктивных утверждений.

|  |  |
| --- | --- |
| ***алг «нахождение минимума»***  ***массив х[1:п]***  ***нач***  ***тп := x[1]***  ***от k = 1 до N цикл***  ***если x[k] < тп то***  ***тп = x[k]***  ***все***  ***кцикл***  ***Min* = *тп***  ***кон*** | Результаты:  mn0 = х[1]  [k = (1 ... N)]    Min = mnN |

**Утверждение.** Для любой последовательности чисел x[l:N] конечным результатом вычислений будет значение Min = Min (х[1], ..., x[N]).

**Доказательство.** Воспользуемся результатами анализа выполнения алгоритма, рассмотренного ранее. Различие между ними состоит в добавлении перед началом цикла присваивания mn := х[1], которое задает начальное значение переменной mn, равное mn0 = х[1].

Тогда в силу приведенных ранее рассуждений и выкладок ко­нечным результатом выполнения новой версии алгоритма будет значение

Min = mnN = Min(x[N], x[N-l], ..., х[2], х[1], mn0) *=*

= Min(x[N], x[N-l], ..., x[2], x[l], x[l]) = Min(x[N], .... х[1]).

**Что и требовалось.**

Рассмотренные примеры являются **образцами доказательств** пра­вильности алгоритмов и программ, которые могут использоваться для анализа и доказательства правильности других новых алгоритмов и программ обработки данных.

Общий вывод, который мы хотим сделать, состоит в том, что применение доказательных методов превращает программирование в **научную дисциплину** создания безошибочных алгоритмов и надеж­ных программ для ЭВМ.

**В о п р о с ы**

1. Как показать наличие ошибок в алгоритме?

2. Сколь долго может продолжаться отладка программ?

3. Зачем нужны доказательства в анализе алгоритмов?

4. Из чего состоит техника доказательств правильности?

5. Когда применяется разбор случаев?

6. Что такое леммы?

7. Что такое индуктивные рассуждения?

**3 а д а ч и**

1. Приведите постановку, алгоритм решения и разбор правильности для следующих задач:

а) подсчет суммы целых чисел;

б) подсчет суммы нечетных чисел;

в) подсчет членов арифметической прогрессии;

г) подсчет членов геометрической прогрессии.

2. Для последовательности чисел **х1, х2,..., хN**, приведите постановку, алгоритм решения и разбор правильности следующих задач:

а) подсчет суммы всех чисел;

б) вычисление среднего арифметического чисел;

в) определение наибольшего из чисел;

г) определение наименьшего из чисел.

3. Для данных о росте и весе учеников приведите постановку задачи, алгоритм решения и разбор правильности для следующих задач:

а) нахождение самого высокого ученика;

г) нахождение самого легкого ученика;

д) нахождение среднего роста учеников;

е) нахождение суммарного веса учеников.

4. Для прямоугольной матрицы **Anm** приведите постановку, алгоритм решения и разбор правильности следующих задач:

а) подсчет сумм элементов матрицы по столбцам;

в) нахождение минимального значения в каждом столбце;

е) нахождение максимального значения в каждой строке;

ж) нахождение наибольшего из минимальных значений в столбцах;

з) нахождение наименьшего из максимальных значений в строках.

5. Для **N** точек на плоскости, заданных случайным образом, при­ведите постановку, метод решения, сценарий, алгоритм и программу решения следующих задач:

а) найти точку, наиболее удаленную от центра координат;

б) соединить пару наиболее удаленных точек;

в) найти три точки, образующие треугольник с наибольшим пери­метром;

г) найти три точки, образующие треугольник с наибольшей пло­щадью.

## 5.5. Решение сложных задач

Большинство практических задач обработки данных относится к числу сложных. **Сложность задач** оценивается сложностью обраба­тываемых данных и сложностью алгоритмов их решения. Сложность данных обычно оценивается их количеством. Сложность алгоритмов оценивается объемом вычислений, необходимых для получения требуемых результатов.

При решении **сложных задач,** требующих составления сложных алгоритмов, особенно сказываются преимущества доказательного программирования. Для этого программы решения сложных задач составляются из вспомогательных алгоритмов и подпрограмм, реша­ющих более простые подзадачи.

**Анализ правильности** сложных алгоритмов и программ распадается на анализ правильности каждого из вспомогательных алгоритмов и на анализ правильности программ в целом. Необходимым условием для этого является составление спецификаций для каждого из вспомо­гательных алгоритмов и каждой подпрограммы,

При таком подходе **доказательство правильности** сложных алго­ритмов и программ подразделяется на доказательство ряда лемм о правильности вспомогательных алгоритмов и подпрограмм и доказательство правильности программ в целом.

В качестве иллюстрации рассмотрим две задачи, которые можно отнести к сложным проблемам обработки данных. Для каждой из этих задач приведем спецификации, алгоритмы и доказательства правильности.

**Первая задача:** упорядочение массивов данных. Пример, для чисел 3, 7, 9, 1, 4 упорядоченная последовательность имеет вид: 1, 3, 4, 7, 9.

Существует несколько способов и методов упорядочения масси­вов и последовательностей. Простейший из них называется методом «пузырька».

**Метод «пузырька»** состоит в нахождении в массиве наименьшего числа и перестановке его на первое место. Это как бы «пузырек», поднимающийся к началу массива. Затем в остатке массива нахо­дится наименьшее число, которое перемещается на второе место, и так далее - до исчерпания всего массива.

Для рассматриваемых чисел метод «пузырька» дает следующие перестановки:

исходные числа: **3**, 7, 9, **1,** 4.

перестановка1: 1, **7**, 9, **3**, 4.

перестановка2: 1, 3, **9,** 7, **4**.

перестановка3: 1, 3, 4, 7, 9. ← **упорядочено.**

Приведем точную математическую постановку задачи.

**Постановка задачи**

Упорядочение последовательности чисел.

Дано: x1, х2, ..., хN - исходные числа.

Треб.: x1', x2', ..., хN' - упорядоченные числа.

Где: х1' ≤ х2' ≤ ... ≤ хN'.

При: N > 0.

Упорядочение чисел по методу «пузырька» в общей форме имеет вид:

**Способ** «упорядочение чисел»

*нач*

*от k=1 до N-1 цикл*

*хтп* := *xk*

*imn := k*

*от i=k+1 до N цикл*

*если xi < хтп то*

*хтп := xi*

*imn* : = *i*

*кесли*

*кцикл*

xmn = Min (хk, ..., хN)

*xk' = хтп*

*ximn ' = xk*

*кцикл* хk′ = Min (хk, ..., хN)

*кон* x1 < х2 < ... < хk′

Приведенный алгоритм можно рассматривать как алгоритм, сло­женный из нескольких фрагментов - вспомогательных алгоритмов, решающих определенные подзадачи.

Первый фрагмент (внутренний цикл) решает подзадачу нахожде­ния минимального значения в подмассиве x[k:N]. Второй фрагмент решает подзадачу перемещения k-го минимального значения на k-e место в массиве.

**Лемма 1.** Для вспомогательного алгоритма

***алг «поиск минимума»***

***нач***

***хтп* := *xk***

***imn := k***

***от i* = *k + 1 до N цикл***

***если* xi < *хтп то***

***хтп* := xi**

***imn := i***

***кесли***

***кцикл*** { xmn = Min (хk, ..., х1) }

***кон***

конечным результатом вычислений будет значение

xmn = Min (хk, ..., хN).

**Доказательство.** Применим индуктивную схему рассуждений. Первое присваивание дает

xmnk = xk.

Далее на первом шаге цикла при i = k + 1 будет получен минимум первых двух чисел:

xk+1 при xk+1 < xmnk,

xmnk+l =

xmnk при xk+1 ≥ xmnk.

На втором шаге цикла будет получен минимум первых трех чисел:

xmnk+2 = min (xk+2, min (хk+1, хk)) = Min (хk+2*,* хk+1, хk).

Теперь можно утверждать, что на третьем и последующих шагах цикла результатом будет минимальное значение среди чисел xk , ..., xi

хmni = Min (хk, ..., хi).

Данное утверждение доказывается с помощью математической индукции. На первых двух шагах при i = k + 1, k + 2 оно уже уста­новлено. Покажем, что оно будет выполняться на (i + 1)-м шаге. Действительно, на следующем шаге цикла результатом будет:

xi+1 при хi+1 < xmni = min(xi+1, хmni)

хmni+1 =

хmni при хi+1 ≥ хmni = min(xi+1, xmni)

= min (xi+1, Min (хk , ..., хi)) = Min (хk, ..., хi, xi+1).

Что и требовалось показать. Следовательно, в силу принципа мате­матической индукции конечным результатом выполнения рассмат­риваемого цикла будет значение:

xmnN = Min (xk, *...,* хN)

**Что и требовалось** доказать.

**Лемма** 2. Для вспомогательного алгоритма

***алг «перестановки»***

***нач*** { xmn = Min (хk, ..., хN) }

***xi′mn= xk***

***кон***

конечным результатом будет значение хk' = Min (хk, ..., хN).

**Доказательство.** В силу леммы 1 xmn = Min (xk, ..., хN). А так как в этом алгоритме хk' *=* xmn, то в итоге получим

хk' = xmn = Min (хk, ..., хN).

**Что и требовалось.**

**Утверждение.** Конечным результатом выполнения алгоритма будет упорядоченная последовательность чисел х1', ..., хN', удовлет­воряющая условию х1' ≤ х2' ≤ ... ≤ хN'.

Доказательство проводится по индуктивной схеме рассуждений. Рассмотрим результаты выполнения основного цикла основного алгоритма:

***алг «упорядочение чисел»***

***нач***

***от k* *=* *1* *до N - 1 цикл***

***xmn*** ***:=*** ***хk***

**...............** { xmn = Min (хk, ..., хi) }

***х′k* = *xmnN***

***хmп′*** = ***хk***

***кцикл*** { хk' = Min (хk, ..., хN) }

***кон*** { х1' ≤ х2' ≤ ... ≤ хk' }

На первом шаге при k = 1 первый элемент последовательности

х1' = Min (x1, х2, ..., хN),

На втором шаге второй элемент последовательности

x2' = Min (х2, ..., хN).

В силу свойств минимума последовательности чисел будем иметь

х1' = Min(x1, x2, ..., хN) = min (x1, Min (х2, ..., хN) ≤ (Min (х2, ..., хN) = x2'.

Таким образом, при k = 2 результатом станут значения х1' и x2', такие что

х1' ≤ x2'

На третьем шаге выполнения основного цикла результатом станет

х3 = Мin(х3, ..., хN).

Опять же в силу свойств минимума последовательности имеем

х2' = Min (х2, х3, ..., хN) = min (x2, Min (x3, ..., хN)) ≤ Min (x3, ..., хN) = x2'.

Таким образом, после третьего шага при k = 3 первые три значе­ния последовательности х1', x2', x3' будут удовлетворять условию

х1'≤ x2'≤ x3'

Из приведенных выкладок можно сделать индуктивное предположение, что на каждом очередном k-м шаге выполнения основного цикла первые k членов последовательности х1', x2', .... хk' будут удов­летворять условию

х1'≤ x2'≤ … ≤ xk'.

Данное предположение доказывается с помощью математической индукции. На начальных шагах при k == 2 и k = 3 оно уже показано. Покажем, что оно будет выполнено на (k + 1)-м шаге, если это усло­вие выполнено на k-м. шаге.

В силу Леммы 2 на k-м и (k + 1)-м шагах выполнения основного цикла промежуточными результатами будут

хk' = Min(xk, xk+1, ..., хN),

хk+1' = Min (xk+1, ..., хN).

В силу свойств минимума последовательности чисел имеем

хk' = Min(xk, xk+1, ..., хN) = min (хk, Min (хk+1, ...,хN)) ≤ Min (xk+1, ..., хN) = хk+1'.

Таким образом, хk ≤ xk+1 и в силу индуктивного предположения получаем, что

x1' ≤ х2' ≤ ... ≤ хk' ≤ xk+'1.

**Что и требовалось** доказать.

Осталось уточнить результаты выполнения последнего шага цикла при k = N - 1. В силу Леммы 2 результатом будет значение

xN-'1 = Min (xN-1, xN) ≤ хN'.

Таким образом, после N - 1 шагов выполнения основного цикла для последовательности в целом будут выполнены соотношения упорядоченности

x1' ≤ x2' ≤ ... ≤ хN' .

**Что и требовалось** доказать. Следовательно, рассмотренный алго­ритм упорядочения чисел правильный в целом.

Применим теперь данный способ упорядочения для решения задачи сортировки. Рассмотрим следующую задачу. Пусть дана не­которая партия товаров с заданной отпускной ценой, указана цена товаров и известны остатки от их продажи. Требуется подсчитать выручку от продажи и отсортировать товары по их остатку.

Данные о товарах представлены двумя таблицами:

**товар стоим кол-во**

|  |  |  |
| --- | --- | --- |
| яблоки | 500 | 200 |
| огурцы | 400 | 250 |
| арбузы | 200 | 600 |

**товар цена остаток**

|  |  |  |
| --- | --- | --- |
| яблоки | 2500 | 100 |
| огурцы | 2000 | 150 |
| арбузы | 1200 | 200 |

Приведем точную постановку задачи и сценарий диалога с ком­пьютером для решения поставленной задачи.

Постановка задачи Сценарий

Сортировка товаров по остатку.

Дано: товары:

D = (d1, d2, .... dN) - данные товара, <товар1> <s1> < m 1> \*

d = (товар, s, m), ...... ... ...

s - стоимость, m - кол-во, остатки:

R = (r1, r2, ..., rN) - данные об остатках, <товap1> <c1> < р1> \*

г = (товар, с, р), ...... ... ...

с - цена, р - остаток.

Треб.: S - сумма выручки, выручка = <S>

R' = (r1', ..., rN') - упорядоченные данные, сортировка:

Где: <товар1'> <с1'> <р1'> \*

S = (c1-s1)⋅(m1-p1) +...+ (сN-sN)⋅(mN-рN), ............

р1' ≤ р2' ≤ ... ≤ рN',

рk' = рi для k = 1 ... N и i = 1 ... N.

При: N > 0.

Для представления исходных данных в программе примем опера­торы **data**:

**tovs: 'товары: osts: 'остатки:**

**data «яблоки», 500, 200 data «яблоки», 2500, 100**

**data «огурцы», 400, 250 data «огурцы», 2000, 150**

**data «арбузы», 200, 600 data «арбузы», 1200, 200**

**data «персик», 800, 100 data «персик», 2000, 0**

**data «», 0, 0 data «», 0, 0**

Приведем теперь алгоритм и программу решения поставленной задачи в соответствии с выбранным сценарием и рассмотренным выше способом упорядочения массивов методом «пузырька».

При составлении алгоритмов и программы решения этой задачи будем использовать принцип **нисходящей разработки** «сверху-вниз»: от основного алгоритма и основной части программы к алгоритмам и подпрограммам решения вспомогательных подзадач.

При решении сложных задач существенным становится органи­зация и представление данных: подбор массивов и переменных для размещения и обработки данных в памяти ЭВМ, а при выделении подпрограмм - процедуры доступа к этим данным.

Для размещения исходных данных о товарах в поставленной задаче примем пять массивов: tv(l:N), s(l:N), m(l:N), с (1:N), p(l:N). Общий размер этих массивов ограничим числом N = 200, которое явно выделено в описании массивов с тем, чтобы в дальней­шем его можно было увеличить для большего количества данных без других изменений программы.

***алг «выручка и остатки товаров»* 'выручка и остатки товаров**

***N* = *100* N = 100**

***массив tv[1:N],s[1:N],m1l:N]* dim tv$(N),s(N),m(N)**

***массив L[1:N],c[1:N],p[1:N]* dim L(N),c(N),p(N)**

***нач* сls**

***вывод («товары:»)* ? «товары:»**

***данные-товаров* gosub tovar 'товары**

***вывод («остатки:»)* ? «остатки:»**

***данные-остатков* gosub ostatok 'остатки**

***вывод («-----»)* ? «-----»**

***подсчет-выручки* gosub vyruch 'выручка**

***вывод («выручка», S)* ? «выручка=»;S**

***вывод («сортировка:»)* ? «сортировка:»**

***сортировка-товаров* gosub sortdan 'сортировка**

***кон***  **end**

По приведенному алгоритму и основной части программы видно, что последовательность ввода-вывода данных о товарах и резуль­татов обработки полностью соответствует выбранному сценарию. Загрузку исходных данных в выбранные массивы в соответствии с принятым представлением выполнят два следующих вспомогательных алгоритма

***алг «данные товаров»* tovar: 'данные товаров**

***нач* '**

***загрузка-товаров* restore tovs**

***от k* = *1 до N цикл* for k = 1 to N**

***чmeнue(tv(k),s(k),m(k))* read tv$(k),s(k),m(k)**

***при tv(k) = «» то* if tv$(k) = «» then exit for**

***вывод (tv(k),s(k),m(k))* ? tv$(k);s(k);m(k)**

***кцикл* next k**

***если k< Nmo N := k-1* if k < N then N = k-1**

***кон* return**

Последний условный оператор изменяет верхнюю границу N массивов в том случае, если фактическое число данных меньше числа мест в массивах, размещенных в памяти компьютера.

***алг «данные остатков»* ostatok: 'данные остатков**

***нач* '**

***загрузка-остатков* restore osts**

***от k = 1 до N цикл* for k = 1 to N**

***чmeнue(tv(k),c(k),p(k))* read tv$(k),c(k),p(k)**

***при tv(k) = «» выход* if tv$(k) = «» then exit for**

***вывод (tv(k),c(k),p(k))* ? tv$(k);c(k);p(k)**

***кцикл*  next k**

***если k < N mo N := k-1* if k < N then N = k-1**

***кон*  return**

Подсчет выручки в соответствии с постановкой задачи по данным, введенным в эти массивы, выполнят следующие вспомогательный алгоритм и подпрограмма:

***алг «подсчет выручки»* vyruch: 'подсчет выручки**

***нач* '**

***S := 0* S = 0**

***от k = 1 до N цикл* for k = 1 to N**

***S := S+(c(k)-s(k)) \*(m(k)-p(k))* S = S+(c(k)-s(k))\*(m(k)-p(k))**

***кцикл*  next k**

***кон*** **return**

**Лемма** **3**. Конечным результатом выполнения данного вспомога­тельного алгоритма будет сумма

SN = (с(1) - s(l))⋅(m(l) - р(1)) + ... + (c(N) - s(N))⋅(m(N) - p(N)).

**Доказательство** проводится с помощью индуктивных рассужде­ний. Первое присваивание S := 0 обеспечивает начальное значение суммы S0 = 0.

О результатах k-го шага выполнения цикла можно сделать индук­тивное утверждение

Sk= Sk-1 + (c(k)*-* s(k))-(m(k) - p(k)) = (с(1) - s(l))⋅(m(l) - p(l)) + ... + (c(k) - s(k))⋅(m(k) - p(k)).

Это утверждение доказывается с помощью математической индукции. На его основании можно сделать заключение о том, что конечным результатом выполнения цикла и алгоритма в целом будет сумма

SN = (с(1) - s(l))⋅(m(l) - р(1)) + ... + (c(N) - s(N))⋅(m(N) - p(N)).

**Что и требовалось** доказать.

Для сортировки данных воспользуемся алгоритмом упорядочения чисел по методу «пузырька», предполагая, что исходная и упорядоченная последовательность чисел r1, r2, ..., rN будут записаны в мас­сиве x[l:N].

Для формирования результирующих упорядоченных данных ис­пользуется массив индексов L(1:N), в котором будут записаны номера элементов исходной последовательности так, что x(k) = p(L(k)) для всех k = 1, ..., N.

***алг «сортировка данных»* sortdan: 'сортировка данных**

***массив x[1:N]* dim x(N)**

***нач '***

***от k = 1 до N цикл* for k = 1 to N**

***L(k) = k* L(k) = k**

***x(k)=p(L(k))* x(k)=p(L(k))**

***кцикл* next k**

***сортировка-массива* gosub sortmas 'сортировка**

***от k = 1 до N цикл* for k = 1 to N**

***i := L(k)* i = L(k)**

***вывод (tv(i),c(i),p(i))* ? tv$(i);c(i);p(i)**

***кцикл* next k**

***кон* return**

Модификация алгоритма упорядочения чисел, размещаемых в массиве x[l:N], с учетом перестановок значений в массиве индексов L[1:N] получает следующий вид:

***алг «сортировка массива»* sortmas: 'сортировка массива**

***нач* '**

***от k = 1 до N-1 цикл* for k = 1 to N-1**

***xmn := x(k)* xmn *=* x(k)**

***imn* := *k* imn = k**

***от i* = *k + 1 до N цикл* for i = k + 1 to N**

***если x(i) < xmn то* if x(i) < xmn then**

***xmn* := *x(i)* xmn = x(i)**

***imn* := *i* imn = i**

***кесли* end if**

***кцикл* next i**

***Imn := L(imn)* Imn = L(imn)**

***xmn* := *x(imn)* xmn = x(imn)**

***L(imn) := L(k)* L(imn) = L(k)**

***x(imn)* := *x(k)* x(imn) = x(k)**

***L(k) :=Imn* L(k) = Imn**

***x(k) := xmn* x(k) = xmn**

***кцикл* next k**

***кон* return**

**Лемма 4.** Результатами выполнения алгоритма сортировки массива будут последовательность чисел, упорядоченная по возрастанию

х(1)' ≤ х(2)' ≤ ... ≤ x(N)'

и последовательность индексов в массиве L[1:N], удовлетворяющих условиям

x(k)' = p(L(k)) для всех k = 1, .... N.

**Доказательство.** Первое утверждение об упорядоченности значе­ний х(1)' ≤ х(2)' ≤... ≤ x(N)' массива x[l:N] по завершении алгоритма следует из доказательства правильности алгоритма упорядочения последовательности чисел. Для доказательства второго утверждения рассмотрим результаты перестановок значений элементов:

***Imn* := *L(imn)*** Imn = L(imn)

***xmn* := *x(imn)*** xmn = x(imn)

***L(imn) := L(k)*** L(imn)' = L(k)

***x(imn) := x(k)*** x(imn)' = x(k)

***L(k) := Imn*** L(k)' = Imn = L(imn)

***x(k) := xmn*** x(k)' = xmn = x(imn)

Перед началом выполнения алгоритма упорядочения массива в алгоритме сортировки данных массив индексов L[1:N] и упорядочи­ваемый массив x[l:N] получают значения, удовлетворяющие следу­ющим соотношениям:

х(i)' = P(L(i) для всех i = 1, ..., N.

Покажем, что эти соотношения сохраняются после каждого шага цикла. Действительно, на каждом очередном k-м шаге цикла будут получены следующие результаты:

Imn = L(imn)

xmn = x(imn) == p(L(imn))

L(imn)' = L(k)

x(imn)' = x(k) = p(L(k)) = p(L(imn)')

L(k)' = Imn = L(imn)

x(k)' = xmn *=* x(imn) = p(L(imn)) = p(L(k)')

Следовательно, после каждого шага цикла для переставленных элементов массивов сохраняются соотношения

x(i)' = p(L(i)) для всех i = 1, ..., N.

**Что и требовалось** доказать.

**Утверждение.** Конечным результатом выполнения алгоритма и подпрограммы сортировки данных будет список данных, в котором последовательность значений р1', р2', ..., рN' будет упорядочена:

p1' ≤ р2' ≤ … ≤ pN'

**Доказательство.** В соответствии с доказанной выше леммой 4 зна­чения в массиве x[l:N] после выполнения алгоритма упорядочения чисел будут удовлетворять условиям

х(1)' ≤ х(2)' ≤ ... ≤ x(N)'.

В силу этой же леммы 4 значения индексов в массиве L[1:N] будут удовлетворять соотношениям x[k]' = p(L(k)) для всех k = 1, ..., N.

Конечным результатом алгоритма сортировки данных является вывод значений из массива p[l:N] в соответствии с массивом индек­сов L[1:N]. Таким образом, очередные значения последовательности p1', p2',... будут равны:

р1' = p(L(l)) = х(1)',

p2'= р(L (2)) = х(2)'и т. д.

В силу упорядоченности значений х(1)', х(2)', ..., x(N)' получаем, что значения выходной последовательности будут также упорядо­чены:

p1' ≤ р2' ≤ … ≤ pN'

**Что и требовалось** доказать.

Следовательно, весь комплекс алгоритмов и подпрограмм пол­ностью соответствует поставленной задаче и гарантирует получение правильных результатов, при любых допустимых исходных данных.

Проверка на ЭВМ программы сортировки товаров, составленной таким систематическим образом, при указанных исходных данных дает следующие результаты:

**товары:**

**яблоки, 500, 200**

**огурцы, 400, 250**

**арбузы, 200, 600**

**персики, 800, 100**

**остатки:**

**яблоки, 2500, 100**

**огурцы, 2000, 150**

**арбузы, 1200, 200**

**персики, 2000, 0**

**выручка = 880000**

**сортировка:**

**персики, 2000, 0**

**яблоки, 2500, 100**

**огурцы, 2000, 150**

**арбузы, 1200, 200**

Таким образом, выполнение программы подтверждает правиль­ность составленного комплекса алгоритмов. Полное и исчерпыва­ющее обоснование их правильности приведено выше.

**В о п р о с ы**

1. Что такое сложные алгоритмы и программы?

2. Что такое упорядоченная последовательность?

3. Что такое упорядочение методом «пузырька»?

4. Как доказывается правильность сложных программ?

5. Что такое разработка программ «сверху-вниз»?

**З а д а ч и**

1. Составьте алгоритм и программу обработки данных о товарах и постройте обоснование их правильности для следующих задач:

а) подсчет планируемых доходов от продажи товаров;

б) подсчет начальной суммы вложений реализации товаров;

в) подсчет планируемой прибыли от продажи товаров;

г) подсчет текущей задолженности.

2. Составьте алгоритм и программу сортировки данных о товарах и постройте обоснование их правильности для следующих задач:

а) сортировка данных по начальному количеству;

б) сортировка данных по остаточному количеству;

в) сортировка данных по начальной стоимости;

г) сортировка данных по продажной цене.

3. Составьте алгоритм и программу сортировки данных о товарах по следующим признакам и приведите обоснование их правильности:

а) по доле планируемых доходов от реализации товаров;

б) по доле прибыли от реализации товаров;

в) по доле убыточности реализации товаров.

# Глава 6. ЭКЗАМЕНЫ ПО ИНФОРМАТИКЕ

## 6.1. Экзамены и зачеты по информатике

Изучение информатики должно заканчиваться **экзаменами,** на которых проверяется знание основ информатики и умения решать задачи на персональных ЭВМ. Зачеты по информатике могут прово­диться по завершении каждого из разделов курса информатики либо в конце курса по совокупности практических заданий.

**Зачеты и экзамены** по информатике могут и должны проводиться с помощью и использованием персональных ЭВМ. При дистанци­онном образовании персональные компьютеры являются основным средством обучения и поэтому экзамены по информатике должны проводиться только с помощью ЭВМ.

На зачетах должны проверяться уровень изучения курса инфор­матики и выполнение компьютерных заданий. **Проверка знаний** и анализ выполнения заданий по информатике могут и должны проверяться на ЭВМ. В качестве средств контроля могут и должны использоваться бумажные копии результатов тестирования и выпол­нения заданий на ЭВМ.

Экзамены в вузах и колледжах, обучающих по программе бакалавриата, служат для проверки знаний студентов в соответствии с **госу­дарственными стандартами** высшего профессионального образования [I], утвержденными правительством Российской Федерации в 1994 г.

В соответствии с государственными стандартами образования все студенты должны освоить технику работы на персональных компью­терах, а также методы и средства накопления, передачи и обработки информации с помощью ЭВМ. Практические вопросы подтвержда­ются выполнением соответствующих заданий на ЭВМ, а теорети­ческие вопросы - **тестированием знаний** с использованием компью­теров.

Для студентов **гуманитарного** и **юридического** профиля дополни­тельно требуются знания принципов представления знаний в ЭВМ и принципов работы систем искусственного интеллекта. Для студен­тов инженерного и экономического профиля дополнительно требу­ются знания основ программирования и решения задач обработки данных на ЭВМ.

На зачетах в соответствии с требованиями государственных стан­дартов должны быть проверены минимальные умения работы на пер­сональных ЭВМ, отвечающих минимальному уровню компьютерной грамотности и выражающихся в выполнениии следующих учебных заданий:

1) оформление на ЭВМ **стихотворения** или юмористического рас­сказа, подготовленных с помощью редактора текстов;

2) оформление на ЭВМ **рекламы** или забавного рисунка, создан­ных с помощью графического редактора;

3) проведение **поиска информации** в сети Интернет по своим лич­ным и профессиональным вопросам и проблемам.

**Базовый уровень** знаний основ информатики и владения средст­вами ЭВМ проверяется на зачетах или экзаменах по результатам самостоятельного выполнения на ЭВМ следующих учебных заданий:

1) организация на ЭВМ **базы данных** о товарах, услугах или фир­мах со своими сведениями в некоторой системе управления базами данных;

2) организация на ЭВМ **базы знаний** о своих знакомых, друзьях или круге предметов с самостоятельно подобранными правилами вывода.

3) организация на ЭВМ **калькуляций и расчетов** закупок товаров или сметы затрат с помощью электронных таблиц.

Для студентов **инженерных** и **экономических** специализаций и направлений бакалавриата дополнительно должны быть проверены знания основ алгоритмизации и программирования, а также умения решать профессиональные задачи с помощью персональных ЭВМ.

Для проверки этого уровня изучения основ алгоритмизации и программирования на зачетах и экзаменах могут быть проверены результаты выполнения следующих учебных задании:

1) организация на ЭВМ диалоговой процедуры или программы с использованием диалоговой системы программирования;

2) организация на ЭВМ обработки данных на основе самосто­ятельно составленных алгоритмов и программ;

3) самостоятельное составление алгоритмов и программ решения задач вплоть до отладки и получения результатов на ЭВМ.

Высшим уровнем изучения основ информатики является овладе­ние технологией решения профессиональных задач с помощью ЭВМ. Это уровень изучения курса информатики проверяется по результа­там выполнения следующих учебных заданий:

1) самостоятельная **постановка задач** и разработка соответству­ющих алгоритмов и программ их решения на ЭВМ:

2) подбор **методов решения** некоторого класса профессиональных задач и его реализации в виде диалоговых программ на ЭВМ;

3) обоснование правильности результатов решения задач, полу­ченных на ЭВМ с помощью самостоятельно созданных алгоритмов и программ.

Экзамены по информатике могут проходить в **устной, письменной** или **компьютерной** форме. Устные и письменные экзамены и зачеты применяются при очной форме обучения. При дистанционном обучении и дистанционном приеме в вузы экзамены и зачеты про­водятся с помощью персональных компьютеров и средств телеком­муникаций.

При традиционной форме устного и письменного экзамена учащимся предлагаются **билеты с вопросами и задачами.** Ответы на вопросы экзаменующиеся излагают устно или письменно соответст­венно форме экзамена. Решения экзаменационных и зачетных за­дач, как правило, излагаются письменно. Примеры вопросов и задач по информатике приведены в предыдущих главах по всему курсу информатики в соответствии с принятыми государственными стан­дартами образования.

При **компьютерной** форме сдачи экзаменов изложение ответов на вопросы и выполнение заданий проводится на компьютере. Письменные ответы оформляются в виде текстовых файлов, а учеб­ные задания - в виде файлов соответствующих инструментальных программных средств - электронных таблиц, баз данных, систем программирования и т. п.

Для **проверки знаний** на зачетах и экзаменах могут применяться тесты. Тестирование знаний является основным средством при дистанционной форме приема зачетов и экзаменов. Тесты могут использоваться в качестве средства проверки знаний и на очных зачетах и экзаменах.

Тесты как средство проверки знаний могут предлагаться в бумаж­ной или компьютерной форме. При **бумажном** тестировании для заполнения тестов используются специальные бланки. Проверка результатов проводится после заполнения бланков. Ответы анализи­руются преподавателями для выведения окончательных оценок.

При **компьютерном** тестировании предварительная оценка отве­тов проводится сразу после ввода их в ЭВМ, а преподаватели выставляют окончательную оценку по протоколам тестирования. Данная форма наиболее удобна для учащихся и существенно упро­щает работу преподавателям.

При **безмашинной форме** зачетов и экзаменов тесты должны служить главным основанием для оценки знаний учащихся. Окончательная оценка определяется исходя из результатов тестирования знаний и результатов выполнения учебных заданий на ЭВМ.

**Ответы** на экзаменах или зачетах должны признаваться **правиль­ными,** если они построены на материалах из учебников или учебных пособий, официально рекомендованных Министерством образо­вания. На экзаменах недопустимы вопросы, выходящие за рамки действующих программ и учебников, утвержденных официальными органами.

При **компьютерной сдаче** экзаменов и зачетов учащиеся демонст­рируют на компьютере свои работы и произведения. Выполненные работы оцениваются по результатам проверки на ЭВМ предложен­ных проектов (баз данных, калькуляций, алгоритмов, программ, гипертекстов и т. д.) путем внесения в них несложных изменений.

**Проверка баз данных** и баз знаний проводится на компьютере поиском информации на запросы и внесением изменений в созданные базы данных и базы знаний. Проверка калькуляций аналогична - получение результатов расчетов и изменение исходных данных в электронных таблицах.

Для **решения задач,** предполагающих составление алгоритмов и программ, может использоваться любой язык и способ описания, изучавшийся в школе или вузе. Описание алгоритмов может прово­диться на псевдокоде, в форме блок-схем или на алгоритмическом языке, изложенном в школьных учебниках по информатике.

Для записи **программ,** могут применяться любые языки програм­мирования - Бейсик, Паскаль, Си, Фортран и т.д. Однако необ­ходимо помнить, что в вузах для обучения и принятия экзаменов используются обычно персональные компьютеры IBM PC с опера­ционной системой MS DOS или Windows.

Программы проверяются на ЭВМ с помощью тестов, предлага­емых преподавателями. Представленные программы оцениваются на **«отлично»,** если они дают правильные результаты на всех контроль­ных тестах. В противном случае оценка зависит от количества и серьезности обнаруженных ошибок.

При безмашинной проверке для оценки предложенных программ используются **спецификации** - сценарии диалога, постановки задач и описания выбранных методов решения. Отсутствие этих специфи­каций может привести к расхождению в оценках правильности пред­ставленных алгоритмов и программ между их авторами и экзамена­торами.

**Исчерпывающим обоснованием** правильности алгоритмов и про­грамм служат соответствующие **доказательства.** Предоставление таких доказательств может сделать излишним проверку программ на ЭВМ и их можно принимать в качестве оснований для отличных оценок за выполненные проекты.

На экзаменах недопустимы задачи, содержание или методы решения которых выходит за рамки **действующей программы** курса информатики. Недопустимо также требовать на экзаменах знание вопросов, отсутствующих в действующих учебниках по информатике.

Во многих вузах **экзамены** по информатике проводятся и для **поступающих.** В 1999 г. приказом № 640 министра образования Рос­сийской Федерации всем вузам разрешено вводить вступительные экзамены по информатике в качестве альтернативных вступительных испытаний на профильные специальности и факультеты.

Для вступительных экзаменов по информатике по заказу Гос­комвуза России в 1994 г. была создана типовая программа [З]. Она основана на учебных программах, утвержденных Министерством образования, и школьных учебниках информатики, имеющихся в средних учебных заведениях.

В 1999 г. более 40 вузов Российской Федерации принимали **всту­пительные экзамены** по информатике: вузы - Москвы, Петербурга, Владивостока, Владимира, Воронежа, Комсомольска-на-Амуре, Перми, Самары, Саратова, Томска, Тулы, Череповца. Полный список вузов, принимающих вступительные экзамены по информатике, можно найти в сети Интернет с помощью запроса «экзамен инфор­матика» в поисковой системе Апорт.

В средних школах **выпускные экзамены** по информатике, как пра­вило, проводятся по выбору учащихся в зависимости от их дальнейших планов. Программы курса информатики с выпускными экзаменами были созданы и рекомендованы Министерством образования для средних школ в 1988, 1992 и 1998 гг. [4, 5].

## 6.2. Решение экзаменационных задач

**Решение задач по информатике** представляют интерес не только для всех студентов, но и для абитуриентов и учащихся средних школ, собирающихся поступать на профильные специальности и факуль­теты. Здесь рассматриваются задачи, предлагавшиеся на вузовских экзаменах по информатике, а также задачи выпускных и вступитель­ных экзаменов в 1994-1997 годах.

На экзаменах по информатике, как правило, включаются **задачи обработки данных** - информационные, логические, экономические, расчетные, комбинаторные и простейшие геометрические задачи. Включение в экзаменационные билеты задач по математике, физике или экономике не рекомендуется, поскольку для их решения требу­ются соответствующие знания, выходящие за рамки курса информа­тики.

Основной **сложностью организации** экзаменов по информатике является необходимость отладки программ и получения результатов на ЭВМ при разнообразии языков программирования - **Бейсик,** Паскаль, Си, Фортран, изучаемых в вузах и школах. В силу этих причин приводимые здесь формулировки задач носят содержательный характер, независимый от языков программирования и используемых ЭВМ.

Основной технической трудностью при решении экзаменацион­ных задач на ЭВМ являются вопросы организации **ввода исходных данных,** имеющих существенные различия в используемых языках программирования, что также отражается на формулировках и усло­виях задач.

Существуют **три основных общих способа** организации ввода исходных данных в персональных ЭВМ, имеющихся в таких языках программирования как Бейсик, Паскаль, Си и Фортран. Рассмотрим их особенности и недостатки.

П е р в ы й с п о с о б - ввод исходных данных с клавиатуры ЭВМ. Этот способ может быть реализован на любых персональных ЭВМ с помощью любого языка программирования. Однако здесь весьма существенен порядок ввода данных, который должен явно указы­ваться в условиях задач.

В т о р о й с п о с о б - запись исходных данных в файлах на маг­нитных дисках. Это способ может быть реализован не на всех пер­сональных ЭВМ и не во всех языках программирования. К тому же не во всех действующих учебниках по информатике имеются примеры решения задач с вводом исходных данных из файлов на магнитных дисках.

Дополнительным **недостатком** этого способа является необходи­мость описания в программах форматов вводимых данных, что пол­ностью отсутствует в учебниках по информатике. Для разрешения этих проблем приходится программировать форматный ввод, что приводит к дополнительным ошибкам как в программах, так и в данных.

Т р е т и й с п о с о б - наиболее удобный для отладки программ на персональных ЭВМ - описание исходных данных внутри текста программ в виде присваивании или операторов **data** на языке Бейсик. Этот способ описания данных приведен в настоящем учебном по­собии, изложен во всех школьных учебниках по информатике и известен всем школьникам, изучавшим информатику в школах.

Однако этот способ, характерный и удобный для диалоговых программ, отсутствует в профессиональных языках программирова­ния таких как Паскаль, Си, Фортран, изучение которых выходит за рамки школьных учебников. По этой причине в формулировках задач по программированию, ориентированных на учащихся с углубленным изучением информатики, используется форматный способ ввода, принятый для професссиональных языков программирования.

Здесь в **примерах программ** решения экзаменационных задач ис­пользуется самый простой и наиболее удобный для отладки программ способ организации ввода тестовых данных в виде операторов data на языке Бейсик. Однако формулировки задач приводятся так, чтобы исходные данные могли вводиться всеми тремя указанными выше способами.

Рассмотрим **образцы решения** экзаменационных задач с приме­рами составления как алгоритмов, так и сценариев диалога. Использо­вание сценариев диалога и является тем средством, которое уравнивает все используемые на персональных ЭВМ языки программирования и позволяет экзаменующимся избегать ошибок ввода-вывода данных, характерных для профессиональных языков программирования.

**Составление сценариев** диалога позволяет до составления алгорит­мов предусмотреть порядок ввода исходных данных и реакции про­грамм на самые различные входные ситуации, которые будут прове­ряться при тестировании на ЭВМ, и тем самым защитить программу и себя от ошибок в исходных данных.

В качестве **основного языка** иллюстраций и примеров программ здесь и далее используется язык Basic для компьютеров IВМ PC как из-за удобств описания входных данных, так и из удобств отладки программ на Бейсике на персональных ЭВМ.

Многолетняя **практика проведения экзаменов** по информатике на ЭВМ показала, что отладка программ на Бейсике стабильно завер­шается на ЭВМ в два раза быстрее, чем на более «мощных» языках, таких как Паскаль, Си или Фортран, что весьма существенно при жестких ограничениях времени на экзаменах.

**Задача 1.** «Информационно-логическая».

Составить алгоритм и программу выбора самого легкого по весу ученика по данным из таблицы, содержащей сведения о фамилиях, именах, росте и весе учеников.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **фамилия** | **имя** | **рост** | **вес** | **пол** |
| Иванов | Вова | 160 | 85 | муж |
| Петрова | Катя | 167 | 67 | жен |
| Сидоров | Миша | 180 | 80 | муж |

Разработку программы решения данной задачи проведем с составления сценария диалога с ЭВМ, что существенно упрощает отладку и работу с программой при решении тестовых задач.

Сценарий

ученики:

<фам> <имя> <вес> <рост> <пол> \*

… … …

самый легкий ученик:

<фам> <имя> <вес>

отсутствует

**Программа Алгоритм**

**' выбор самого легкого ученика *алг «выбор самого легкого ученика»***

**сls *' нач***

**? «ученики:» *' вывод («ученики:»)***

**vs = 0 *' vs = 0***

**do *' цикл***

**read fm$, nm$, r, v, pl$ *' ввод fmS, nm$, r, v, pl$***

**if fm$ = «» then exit do *' если fm$ = «» то выход***

**fm$, nm$, r, v, pl$ *' вывод fm$, nm$, r, v, pl$***

**if р1$=»муж» then *' если pl$= «мyж» то***

**if vs = 0 then *' если vs = 0 то***

**vs = v *' vs = v***

**fs$ = finS: ns$ = nm$ *' fs$ = fin$: ns$ = nm$***

**elseif v < vs then *' инес v < vs то***

**vs = v *' vs = v***

**fs$ = fm$: ns$ = nm$ *' fs$ =fm$: ns$ = nm$***

**end if *' кесли***

**end if *' кесли***

**loop *' кцикл***

**? «самый легкий ученик:» *' вывод («самый легкий ученик:»)***

**if vs = 0 then *' если vs = 0 то***

**? «отсутствует» *' вывод («отсутствует»)***

**elseif vs > 0 then *' инес vs > 0 то***

**? fs$, ns, vs *' вывод (fs$, ns, vs)***

**end if *' кесли***

**end *' кон***

**data «Иванов», «Вова», 160, 85, «муж»**

**data «Петрова», «Катя», 167, 67, «жен»**

**data «Сидоров», «Миша», 180, 80, «муж»**

**data «», «», 0, 0, «»**

Отметим, что при использовании языка Basic тексты программ и описания алгоритмов полностью идентичны друг другу и по форме и по содержанию. Можно сказать, что текст программы на Бейсике получается переводом русских слов и словосочетаний на язык Бейсик и наоборот.

**Задача 2.** «Экономическая».

Составить алгоритм и программу определения общей стоимости промышленных товаров по данным из таблицы:

|  |  |  |  |
| --- | --- | --- | --- |
| **товар** | **тип** | **цена** | **кол-во** |
| ананасы | прод | 8000 | 40 |
| утюги | пром | 60000 | 3 |
| сахар | прод | 6000 | 20 |

Разработку алгоритма и программы начнем с составления сцена­рия диалога, учитывая возможность отсутствия в таблице требуемых исходных данных.

Сценарий

промышленные товары

отсутствуют

<товар> <цена> <кол> <стоим> \*

… … …

общая стоимость = <sum>

**Программа**  **Алгоритм**

**' стоимость промтоваров *'* *алг «стоимость промтоваров»***

**сls ' *нач***

**? «промтовары:» *' вывод («промтовары:»)***

**n = 0: sum = 0 ' *п* = *0: sum* = *0***

**do ' *цикл***

**read tv$, tp$, сn, kl ' *ввод tv$, tp$, сn, kl***

**if tv$ = «» then exit do ' *если tv$* = *«» то выход***

**if tp$ = «пром» then ' *если tp$ = «пром» то***

**n = n + 1 ' *n =n + 1***

**st = cn\*kl ' *st = cn \*kl***

**? tv$, en; kl; st ' *вывод (tv$, en, kl, st)***

**sum = sum + st ' *sum* = *sum* + *st***

**end if *' кесли***

**loop *' кцикл***

**if n = 0 then ' *если n = 0 то***

**? «отсутствуют» *' вывод («отсутствуют»)***

**else *' иначе***

**? «общая cтoимocть=»,sum ' *вывод(«общая стоимость*=», *sum)***

**end if ' *кесли***

**end *' кон***

**data «сахар», «прод», 6000, 20**

**data «утюги», «пром», 60000, 3**

**data «книги», «пром», 4000, 30**

**data «», «», 0, 0**

Рассмотрим в качестве иллюстрации **примеры решения** экзаменаци­онных задач в МЭСИ - Московском государственном университете экономики, статистики и информатики. Этот университет одним из первых в 1991 году ввел вступительные экзамены по информатике и стал лидером в дистанционном образовании среди государственных вузов Российской Федерации.

Задание **на экзаменах в МЭСИ** состоит из пяти задач. Первая задача по системам счисления. Вторая задача - на алгебру логики. Третья задача - тест или анализ блок-схемы. Четвертая и пятая задача - задача на составление алгоритмов и программ.

Первые три задачи в экзаменационных билетах МЭСИ по слож­ности оцениваются на два балла, а четвертая и пятая задача - на четыре и пять баллов. Положительную оценку на экзамене получает та работа, в которой набрано не менее 8 баллов.

Таким образом **подсчет баллов** показывает, что в МЭСИ для по­лучения положительной оценки на экзаменах по информатике необходимо решить хотя бы одну задачу на составление программ, а решение задач на составление двух программ - гарантирует на экзамене положительную оценку.

В виду указанной особенности вступительных экзаменов по ин­форматике в МЭСИ разберем примеры решения задач на составле­ние программ, используя для описания алгоритмов псевдокод, а не блок-схемы, как это делается в учебниках МЭСИ.

**Задача 1.** Написать программу на любом языке программирова­ния согласно следующему условию.

Дана целочисленная матрица А размера M×N, где M,N - задан­ные натуральные числа. Найти количество столбцов матрицы, со­держащих одни нулевые элементы.

Пример матрицы:



Для представления матрицы в программе на языке Бейсик можно использовать операторы data, в первой строке которых указывается размерность матрицы:

**data 5**

**data 1, 0, 1, 0, 0**

**data 0, 1, 0, 0, 0**

**data 0, 0, 1, 0, 0**

**data 0, 1, 0, 0, 0**

**data 0, 0, 0, 0, 0**

Для вывода исходных данных и результатов их обработки можно воспользоваться следующим сценарием:

**Матрица А<n><n>:**

**<а11> ... <a1n>**

**… … …**

**<anl> … <ann>**

**Число нулей в столбцах:**

**<d1> ... <dn>**

Решением поставленной задачи на ЭВМ можно получить с помо­щью следующего алгоритма и программа на языке Бейсик. Обратите внимание в программе используются массивы переменной длины, которая определяется при вводе размеров матрицы А:

**' подсчет нулевых столбцов ' *алг «подсчет нулевых столбцов»***

**' в квадратной матрице Ann ' *нач***

**read n ' *чтение(п)***

**dim A(n,n), D(n) *' массивы А(1:п,1:п), D(1:n)***

**print «Матрица A»;n;n;«:» *' вывод («Матрица А»;п;п;* *«:»)***

**for k = 1 to n *'* *от k* = *1 до п цикл***

**for 1 =1 to n *' от l =1 до п цикл***

**read A(k,l) *' чтение A(k,l)***

**print A(k,l) *'* *вывод A(k,l)***

**next 1 *' кцикл***

**next k *' кцикл***

**for k = 1 to n *' om k= 1 до п цикл***

**D(k) *=* 0 ' *D(k)* = *0***

**for 1 = 1 to n ' *от l=1 до п цикл***

**if A(k, l) = 0 then ' *если A(k, l)* = *0 то***

**D(k) = D(k) + 1 *' D(k)* = *D(k) + 1***

**end if ' *кесли***

**next 1 ' *кцикл***

**print D(k); ' *вывод D(k);***

**next k ' *кцикл***

**end ' *кон***

**Задача** 2. Дана строка символов. Распечатать все слова нечетной длины, отличные от второго слова.

(В этой задаче **«словом»** называется группа символов, разделен­ная с одной или обеих сторон одним или несколькими пробелами и не содержащую внутри себя пробелов.)

Пример строки

**Я волком бы выгрыз бюрократизм.**

**К мандатам почтения нет.**

Результат обработки

**бы**

**выгрыз**

**бюрократизм.**

**почтения**

**нет.**

Для представления строк в программе на Бейсик можно восполь­зоваться операторами **data:**

**data** «Я волком бы выгрыз бюрократизм.»

**data** «К мандатам почтения нет.»

**data «»**

Здесь пустое слово «» означает конец исходного текста.

Для вывода исходных данных и результатов их обработки можно принять следующий сценарий:

**исходный текст:**

**<строка1>**

**… …**

**<строкаn>**

**слова нечетной длины:**

**<слово1>**

**… …**

**<словоm>**

Решение поставленной задачи на ЭВМ можно получить с помо­щью следующих алгоритма и программы на Бейсике, в которых в виде вспомогательного алгоритма и подпрограммы выделена обра­ботка каждой отдельной строки текста:

**' выделение слов нечетной длины ' *алг «слова нечетной длины»***

**print «исходный текст:» ' *вывод «исходный текст;»***

**n = 0: s2$ = «» *' n = 0: s2$ = «»***

**print «исходный текст:» *' вывод «исходный текст:»***

**do '*цикл***

**read str$ *' чтение\_строки***

**if str$ = «» then exit do ' *при str$* = *«» выход***

**print str$ ' *вывод\_строки***

**gosub stroka *' обработка\_строки***

**loop *' кцикл***

**end ' *кон***

**stroka: ' обработка строки ' *алг «обработка строки»***

**dl = len(sfr$) ' *dl = длuнa(str$)***

**print «слова нечетной длины:» ' *вывод «слова нечетной длины:»***

**sl = 0 *' sl=0***

**for k=l to dl *' от k = 1 до dl цикл***

**if str$(k) 0 «» then *' если str$(k) ≠ «» то***

**sl = sl + 1 ' *sl = sl + 1***

**elseif sl > 0 then ' *инеc sl > 0 то***

**p = k - sl + 1 ' *p = k - sl + 1***

**slv$ = mid$(str$,p,sl) ' *slv$* = *cpeдн.(str$,p,sl)***

**n = n + 1 ' *n = n + 1***

**if n = 2 then ' *если n = 2 то***

**sl2$ = slv$ *' sl2$ = slv$***

**elseif slv$ 0 sl2$ then *' инеc slv$ ^ sl2$ то***

**if (sl/2)\*2= si then *' если (sl/2) \*2 = sl то***

**print slv$ *' вывод slv$***

**end if *' кесли***

**end if *' кесли***

**sl = 0 *' sl = 0***

**end if *' кесли***

**next k *' кцикл***

**return *' кон***

**Экзаменационные задачи МЭСИ (Московский государственный**

**университет экономики, статистики и информатики)**

1. Дана действительная квадратная матрица А порядка N, где N - заданное натуральное число, все элементы которой различны. Сколько элементов матрицы равны (МАХ + MIN)/2, где МАХ, MIN - соответственно, максимальное и минимальное значения среди элементов матрицы.

2. Дана целочисленная матрица А размера M×N, где М, N - заданные натуральные числа. Сформировать одномерный массив В, где B(i) равно сумме элементов, кратных пяти и расположенных в i строке матрицы i = 1,2, .... М.

3. Дана целочисленная матрица А размера MxN, где М, N - заданные натуральные числа. Найти количество столбцов матрицы, содержащих одни нулевые элементы

4. Дана квадратная целочисленная матрица А порядка N, где N - заданное натуральное число, все элементы которой различны. По­менять местами строку, в которой находится наименьший элемент матрицы, со строкой, где находится наибольший элемент матрицы.

5. Дана целочисленная матрица А размера М х N, где М, N - заданные натуральные числа, причем М > 5. Найти количество столб­цов матрицы, в каждом из которых содержится не менее 5 нулевых столбцов.

6. Дана квадратная целочисленная матрица А порядка N, где N - заданное натуральное число. Является ли заданная матрица маги­ческим квадратом, т. е. такой матрицей, в которой суммы элементов во всех строках и столбцах одинаковы

7. Дана действительная матрица А размера M×N, где М, N - заданные натуральные числа, все элементы которой различны. Сформировать одномерный целочисленный массив В, где B(j) равно среднему арифметическому значению индексов наибольшего и наи­меньшего элементов в j -ом столбце j =1,2, .... N.

8. Дана строка символов. Распечатать все слова с количеством символов больше 4 и меньше 10.

9. Дана строка символов. Распечатать самое длинное слово, начинающееся на букву «К».

10. Дана строка символов. Распечатать самое длинное слово, первые две буквы которого «КО».

11. Дана строка символов. Составить одномерный массив из слов, которые отличны от слова INFORMATION.

12. Дана строка символов. Распечатать самое длинное симмет­ричное слово, первые две буквы которого «КО».

13. Дана строка символов. Выяснить, какое слово встречается раньше в строке с наименьшим или наибольшим количеством симво­лов.

14. Дана строка символов. Определить среднее количество сим­волов в словах четной длины.

15. Дана строка символов. Распечатать все слова нечетной длины, начинающиеся и оканчивающиеся на букву «Т».

## 6.3. Проверка программ на ЭВМ

В экзаменационных задачах и заданиях проверка правильности составленных программ проводится на ЭВМ путем их испытания на специально подбираемых тестах. Эта работа проводится препода­вателями или экзаменаторами по завершении отладки программ на ЭВМ.

Напомним два основных определения, на которых базируется оценка правильности программ по результатам их тестирования:

1) программа объявляется **содержащей ошибки,** если можно ука­зать тесты, при которых выполнение программы на ЭВМ приводит к отказу, сбою или получению неправильных результатов;

2) программа является **правильной,** если при любых допустимых исходных данных она дает правильные результаты.

Напомним два дополнительных, но весьма существенных опре­деления допустимости исходных данных:

1) исходные данные считаются **допустимыми,** если для этих данных существует решение поставленных задач;

2) исходные данные **недопустимы,** если для этих данных постав­ленная задача не имеет решений.

Наконец правильность результатов решения регламентируется следующими двумя определениями:

1) результаты решения **правильные,** если они соответствуют требованиям поставленной задачи;

2) результаты решения **неправильные,** если они противоречат требованиям поставленной задачи.

Подбор тестов для проверки программ опирается на некоторые общие принципы, среди которых можно выделить следующие:

1) проверка **частных случаев** задачи;

2) проверка **основных случаев** задачи;

3) проверка **граничных случаев.**

**Частные случаи** - те, которые допускают простейшие способы решения и проверки правильности результатов. Граничными случаями объявляются данные, которые лежат на границе между допустимыми и недопустимыми данными.

Разбор этих понятий проведем на примере типичной экзамена­ционной задачи по информатике.

**Задача.** «Средняя зарплата».

Среди N сотрудников отдела выделить тех, кто получает зарплату выше средней по данным из следующей таблицы:

|  |  |  |
| --- | --- | --- |
| **Иванов** | **начальник** | **2500** |
| **Петров** | **сотрудник** | **1800** |
| **Сидоров** | **секретарь** | **900** |

Типичность этой задачи заключается в том, что она является одной из **задач обработки** **данных**, представленных в нашем случае таблицей. Частным случаем в этой задаче является таблица, состо­ящая из одной строки (случай N = 1). Граничным случаем - ситу­ация, когда все сотрудники получают одинаковую зарплату и никто из них не получает зарплату выше средней.

**Типичными ошибками** в программах решения этой задачи может быть отсутствие обработки частного случая, когда N = 1, либо отсутст­вие ответа на граничную ситуацию, когда все получают одинаковую зарплату.

Приведем примеры тестов, применявшихся при проверке на ЭВМ правильности программ решения этой задачи:

1) **тест1** (проверка основного случая):

|  |  |  |
| --- | --- | --- |
| **Иванов** | **начальник** | **2500** |
| **Сидоров** | **секретарь** | **900** |

2) **тест2** (проверка частного случая):

|  |  |  |
| --- | --- | --- |
| **Иванов** | **начальник** | **2500** |

3) **тест3** (проверка граничных ситуаций):

|  |  |  |
| --- | --- | --- |
| **Иванов** | **начальник** | **2000** |
| **Петров** | **сотрудник** | **2000** |

Независимо от языка программирования и способа ввода-вывода данных отсутствие каких бы то ни было ответов - есть «отказ», вывод избыточных данных или неполные данные - есть «сбой», непра­вильный список - есть неправильный результат решения.

Правильными ответами с точностью до формулировок в этой задаче являются:

**ответ1** (основной случай):

|  |  |  |
| --- | --- | --- |
| **Иванов** | **начальник** | **2500** |

**ответ2** (частный случай):

**«никто не получает зарплату выше средней»**

**ответ3** (граничный случай):

**«никто не получает зарплату выше средней»**

Приведем правильное решение поставленной задачи в форме про­граммы на языке Basic, в которой исходные данные записываются в списке операторов data. Разработку приводимой ниже программы проведем, начиная с составления сценария, в котором учитываются все ситуации и частные случаи постановки задачи.

Сценарий

список сотрудников:

<фамилия> <долж> <з/плата> \*

… … …

средняя з/плата = <среднее>

з/плата выше средней:

<фамилия> <з/плата> \*

… … …

не получает никто

**Программа Алгоритм**

**' выше средней з/платы *алг «выше средней з/платы***

**сls *нач***

**? «сотрудники:» *вывод «сотрудники:»***

**do *цикл***

**read Hn$, dl$, zp *чтение fm$, dl$, zp***

**if fm$=«» then exit do *если fm$=«» то выход***

**? fm$, dl$, zp *вывод fm$, dl$, zp***

**sum *=* sum + zp *sum = sum + zp***

**n = n + 1 *n* *= n + 1***

**loop *кцикл***

**sr = sum/n *sr = sum/n***

**? «средняя з/плата=»; sr *вывод («средняя 3/nлama=»;sr)***

**? «з/плата выше средней:» *вывод(«з/плата выше средней:»)***

**restore zplts *перезагрузка данных***

**ns = 0 *ns = 0***

**for k = 1 to n *от k = 1 до п***

**read fm$, dl$, zp *чтение fm$, dl$, zp***

**if zp > sr then *если zp > sr то***

**? fni$, zp *вывод (fm$, zp)***

**ns = ns + 1 *ns = ns + 1***

**end if *кесли***

**next k *кцикл***

**if ns = 0 then *если ns* = *0 то***

**? «нe получает никто» *вывод («не получает никто»)***

**end if *кесли***

**end *кон***

**zplts: 'данные о зарплате:**

**data «Иванов», «начальник», 250000**

**data «Сидоров», «секретарь», 90000**

**data «», «», 0**

Проверку правильности этой программы следует провести на указанных выше трех тестах. Достоинства приведенной программы:

1) хорошо организованный вывод результатов, совмещенный с выводом исходных данных (свойства, заложенные в сценарий);

2) удобная организация структуры программы, позволяющая локализовать возможные ошибки (свойства структурированных алгоритмов);

3) удобная организация исходных данных в тексте программы, позволяющая упростить процедуру тестирования и отладки программы на ЭВМ.

**Экзаменационные задачи ГУУ**

**(Государственный университет управления)**

**Задача 1**

Каждое из N фермерских хозяйств представило свой перечень из М машин разных наименований (марок) на их приобретение в единст­венном экземпляре (N и М заданы). Составить общий перечень необходимых марок машин с указанием их количества, расположив марки в порядке убывания потребности в них.

**Задача 2**

N сотрудников (известны фамилии) работают в 2 смены по инди­видуальному графику (1-й день - «утро», 2-й день - «вечер», 3-й день - «выходной»). Все они в свое нерабочее время должны пройти диспансеризацию в медпункте, который работает ежедневно в 2 смены. В день начала диспансеризации о каждом сотруднике известно в какую смену он работает или то, что он выходной. Со­ставить ежедневные списки посещения сотрудниками медпункта с указанием времени посещения («утро» и «вечер»), учитывая, что в каждой смене медпункта могут быть приняты не более М человек должен посетить медпункт один раз. Числа N и М заданы.

**Задача 3**

На кинофестивале 35 стран представили свои фильмы. Общее число фильмов не превышает 100. Известны названия стран - участ­ниц и фильмов, а также баллы, полученные каждым из фильмов. Определить фильм, завоевавший первый приз (максимальный балл) и страну, получившую наибольший средний балл за представленные фильмы. Считать, что фильмы в общем списке по странам не упоря­дочены, а фильм и страна, его представляющая, является единствен­ными победителями.

**Задача 4**

Известны очки, полученные каждым из М спортсменов-много­борцев в каждом из N видов соревнований (N и М заданы). Для каждого из спортсменов определить, в каких видах соревнований он получил результат не хуже других спортсменов и какой конкретно. Фамилия спортсменов и названия видов соревнований известны.

**Задача 5**

Даны сведения о соревновании N фигуристов ( N - заданное число): фамилия, наименование спортивного общества, 10 оценок за выступление. Требуется по каждому спортивному обществу опре­делить фигуриста, показавшего наивысший результат, считая его единственным. Баллы, полученные фигуристом, подсчитываются следующим образом: максимальная и минимальная оценки отбра­сываются, а из остальных формируется средняя.

## 6.4. Олимпиадные задачи по информатике

Особый интерес у студентов и школьников, увлекающихся ин­форматикой, вызывают олимпиадные задачи - наиболее сложные задачи из курса информатики, с помощью которых в форме сорев­нования выявляются наиболее талантливые и способные учащиеся.

Согласно приказу министра образования Российской Федерации № 500 победители и призеры международных олимпиад могут руко­водством российских вузов зачисляться **без экзаменов** на профиль­ные специальности и факультеты.

Победителям и призерам российских и региональных олимпиад ректора вузов победы в таких олимпиадах согласно указанному при­казу могут засчитывать как **успешную сдачу** профильных вступитель­ных экзаменов.

Особенностью олимпиад по информатике является то, что решение олимпиадных задач и выполнение конкурсных заданий проводится **исключительно на ЭВМ**. Второй особенностью олимпиад по инфор­матике в силу использования персональных компьютеров является форма проведения олимпиад.

В 1995 году по инициативе **Международной академии информати­зации** была проведена первая сетевая олимпиада, в которой приняло участие более 200 учащихся Москвы и Московской области. Нова­цией этой олимпиады было то, что задачи и результаты их решения передавались с помощью электронной почты, а оценка составленных программ проводилась на ЭВМ с использованием заранее подготов­ленных тестов.

Победителям и призерам этой олимпиады, решившим наиболь­шее число задач с наименьшим числом ошибок, было предложено поступление без экзаменов в Московский институт электроники и математики (МИЭИ) для обучения по специальностям в области информатики и вычислительной техники.

Примеры олимпиадных задач по информатике в других уни­верситетах и вузах Российской Федерации, которые засчитывают результаты побед в региональных, российских и международных олимпиадах по информатике, можно найти в Интернете по запросу «олимпиада информатики» с помощью поисковых систем Апорт, Ремблер или Яндекс. В 1999 году таких вузов было более сорока.

Ниже приводятся тексты задач первого тура первой сетевой олим­пиады с указанием максимального числа баллов за решение этих задач, а также примеры программ их решения на языке Basic.

**Оценки за решение** задач проставлялись по следующей методике:

1) при правильных результатах на всех тестах 100% баллов; 2) при получении правильного решения хотя бы на одном тесте 40% баллов, а за результаты на остальных (n - 1 )-м тестах добавляется 60%/(n - 1) баллов; 3) при неправильных результатах на всех тестах или отсутст­вии программы оценка не ставилась.

На первом туре первой сетевой олимпиады были предложены четыре задачи информационно-логического и геометрического со­держания со следующими оценками сложности, определенными экспертами:

**задача 1 («Экзамены»)** - 50 баллов;

**задача 2 («Слова»)** - 100 баллов;

**задача 3 («4 точки»)** -150 баллов;

**задача 4 («Ломаная»)** - 250 баллов.

Более 120 участников из 200 представили решения задач. Из них более 20 представили решения трех задач, девять участников пред­ложили решения четырех задач. Правильное решение четырех задач представил только один участник, но даже и у него в последней четвертой задаче программа не прошла все тесты.

В целом задачи были подобраны по принципу **от простого к слож­ному**. С одной стороны это дало всем успевающим в информатике ученикам довести до успешных результатов хотя бы одну программу, а с другой стороны - сложность и дифференциация задач были таковы, чтобы можно было увидеть уровень подготовки и оценить способности участников.

Рассмотрим формулировки задач, проверочные тесты и правильные решения в форме программ на языке Basic. Первая задача относится к классу информационно-логических.

**Задача 1**. «Экзамены».

Среди N абитуриентов, сдававших экзамены по информатике, математике и языку, выбрать всех отличников и всех учащихся, на­бравших в сумме не меньше проходного балла. Данные о проходном балле вводятся с клавиатуры, а данные о результатах сдачи экзаме­нов представлены таблицей:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **фамилия** | **имя** | **информатика** | **математика** | **язык** |
| Иванов | Саша | 4 | 4 | 3 |
| Петрова | Катя | 5 | 5 | 5 |
| Сидоров | Алеша | 5 | 3 | 3 |

Приведем проверочные тесты и правильные результаты:

**Тест 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Иванов | Саша | 4 | 4 | 3 |
| Петрова | Катя | 5 | 5 | 5 |
| Сидоров | Алеша | 5 | 3 | 3 |

проходной балл =? 12

**Правильные результаты:**

отличники:

Петрова Катя

не меньше проходного:

Иванов Саша

Петрова Катя

**Тест 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Иванов | Саша | 4 | 4 | 3 |
| Сидоров | Алеша | 5 | 3 | 3 |

проходной балл =? 12

**Правильные результаты:**

отличники:

отсутствуют

не меньше проходного:

Иванов Саша 4 4 4

**Тест 3:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сидоров | Алеша | 5 | 3 | 3 |

проходной балл =? 14

**Правильные результаты:**

отличники:

отсутствуют

не меньше проходного:

отсутствуют.

В приведенных тестах анализируются различные логические ситуации с отсутствием «отличников» или «успешно» сдавших экза­мены. При составлении программы эти ситуации можно явно преду­смотреть в сценарии диалога с ЭВМ:

Сценарий

оценки учащихся:

<фам> <имя> <мат> <инф> <язык> \*

………………………………….

проходной балл=? <b1>

отличники:

<фам> <имя> \*

……………

отсутствуют

не меньше проходного:

<фам> <имя> <sum> \*

……………..

отсутствуют

**Программа Алгоритм**

**' результаты экзаменов *алг «результаты экзаменов»***

**cls *нач***

**? «оценки учащихся:» *вывод («оценки учащихся:»)***

**do *цикл***

**read fm$, nm$, mt, in, zk *ввод fm$, nm$, mt, in, zk***

**if fm$ = «» then exit do *если fm$ = «» то выход***

**? fm$, nm$, mt, in, zk *вывод (fm$, nm$, mt, in, zk)***

**loop *кцикл***

**input «проходной балл=»,b1 *запрос («проходной балл=»,b1)***

**restore ocenki *перезагрузка\_ oценки***

**? «отличники:» *вывод («отличники:»)***

**n = 0 *п = 0***

**do *цикл***

**read fm$, nm$, mt, in, zk *ввод fm$, nm$, mt, in, zk***

**if fm$ = «» then exit do *если fm$ = «» то выход***

**if mt=5 and in=5 and zk=5 then *если mt=5 и* *in = 5* *и zk=5 то***

**? fin$, nm$ *вывод (fm$, nm$)***

**n = n + 1 *n* = *n + 1***

**end if *кесли***

**loop *кцикл***

**if n=0 then ? «отсутствуют» *если п=0 то вывод(«отсутствуют»)***

**restore ocenki *перезагрузка-оценок***

**? «не меньше проходного:» *вывод («не меньше проходного:»)***

**n = 0 *п = 0***

**do *цикл***

**read fm$, nm$, mt, in, zk *ввод fm$, nm$, mt, in, zk***

**if fm$ = «» then exit do *если fm$ = «» то выход***

**sum = mt + in + zk *sum = mt + in + zk***

**if sum >= hi then *если sum* >= *bl то***

**? fm$, nm$, sum *вывод (fm$, nm$, sum)***

**n = n + 1 *n* = *n + 1***

**end if *кесли***

**loop *кцикл***

**if n = 0 then ? «отсутствуют» *если п = 0 то вывод («отсутствуют»)***

**end *кон***

**ocenki: 'оценки учащихся**

**data «Иванов», «Саша», 4, 4, 3**

**data «Петрова», «Катя», 5, 5, 5**

**data «Сидоров», «Алеша», 5, 3, 3**

**data «», «», 0, 0, 0**

Рассмотренная задача имеет чисто квалификационный характер проверки знаний информатики по школьной программе и умения самостоятельно составлять алгоритмы и программы решения на ЭВМ простейших информационных задач. С этой задачей справилось большинство участников олимпиады. Однако далеко не все преду­смотрели исключительные ситуации и в результате многие из них потеряли определенную часть баллов на указанных тестах.

Вторая олимпиадная задача также относится к классу информа­ционно-логических задач. Ее содержание заключается в переработке символьных данных.

**Задача** 2. «Слова».

Для фразы на русском языке, в которой нет знаков препинания, а слова отделяются одним единственным пробелом, организовать циклическую перестановку слов.

Исходная фраза:

ВЕЧЕРАМИ МЫ СМОТРИМ ТЕЛЕВИЗОР

Циклическая перестановка слов:

МЫ СМОТРИМ ТЕЛЕВИЗОР ВЕЧЕРАМИ

СМОТРИМ ТЕЛЕВИЗОР ВЕЧЕРАМИ МЫ

ТЕЛЕВИЗОР ВЕЧЕРАМИ МЫ СМОТРИМ

ВЕЧЕРАМИ МЫ СМОТРИМ ТЕЛЕВИЗОР

Сценарий

Исходная фраза:

<строка>

Перестановка слов:

<строка'> \*

Проверочные .тесты:

**Тест 1:** Исходная фраза:

утром был дождь

**Правильные результаты:**

Перестановка слов:

был дождь утром

дождь утром был

утром был дождь

**Тест** 2: Исходная фраза:

правильно

**Правильные результаты:**

Перестановка слов:

правильно

**Программа Алгоритм**

**′ перестановка слов *алг «перестановка слов»***

**cls *нач***

**? «Исходная фраза:» *вывод («Исходная фраза:»)***

**line input st$ *ввод-строки (st$)***

**? st$ *вывод st$***

**In = len(st$) *in* = *len(st$)***

**? «Перестановка слов:» *вывод («Перестановка слов:»)***

**s$ = st$ *s$ = st$***

**do *цикл***

**k = instr(s$,«») *k* = *instr(s$,«»)***

**if k = 0 then *если k = 0 то***

**? s$ *вывод (s$)***

**exit do *выход***

**end if *кесли***

**lf$ = left$(s$,k-l) *lf$* = *left$(s$,k-l)***

**rt$ = right(s$,ln-k) *rt$* = *right(s$,ln-k)***

**ns$ = rt$ + «» + lf$ *ns$* = *rt$* + *«» + lf$***

**? ns$ вывод *(ns$ )***

**if ns$ = st$ then exit do *при ns$* = *st$ выход***

**s$ = ns$ *s$* = *ns$***

**loop *кцикл***

**end *кон***

Третью задачу можно отнести к числу комбинаторных задач, реше­ние которых заключается в организации перебора различных вари­антов данных.

**Задача 3**.«4 точки».

Для заданных четырех точек на плоскости найти длину мини­мального и максимального обхода их по замкнутому маршруту. Дан­ные о координатах точек представлены в таблице:

|  |  |
| --- | --- |
| **х** | **у** |
| 0 | 0 |
| 0 | 3 |
| 4 | 0 |
| 5 | 10 |

Составление алгоритмов и программы для решения этой задачи также полезно начать с составления сценария диалога.

Сценарий

координаты точек:

<х1> <у1>

… … …

<х4> <у4>

максимальный маршрут:

<ml> <m2> <m3> <m4>

длина = <mх>

минимальный маршрут:

<n1> <n2> <n3> <n4>

длина = <mn>

Простейший способ решения этой задачи заключается в орга­низации перебора всех замкнутых маршрутов, проходящих через заданные точки и выбора среди минимального и максимального по длине маршрутов.

**Программа**  **Алгоритм**

**′мин. и макс. маршруты *алг «мин. и макс. маршруты»***

**cls *нач***

**n = 4 *п* = *4***

**dim x(n),y(n),r(n,n) *dim x(n),y(n),r(n,n)***

**? «координаты точек» *вывод («координаты точек»)***

**gosub vvdan 'ввод данных *ввод-координат-точек***

**restore mrshrt 'маршруты *загрузка-маршрутов***

**? «маршруты:» *вывод («маршруты:»)***

**mr = 1\*2\*3 *mr =1\*2\*3***

**mx = 0 *тх = 0***

**for l = 1 to mr *от l* = *1 до mr***

**read k1, k2, k3, k4 *ввод k1, k2, k3, k4***

**dl = r(kl,k2) + r(k2,k3) *dl* = *r(kl,k2)* + *r(k2,k3)***

**d3 = r(k3,k4) + r(k4,kl) *d3 = r(k3,k4)* + *r(k4,k1)***

**d = dl + d3 *d* = *d1 + d3***

**? kl; k2; k3; k4, d *вывод (k1; k2; k3; k4, d)***

**if mx = 0 then *если тх* = *0 то***

**mx = d: mn *=* d *mx* = *d: mn = d***

**ml *=* kl: m2 = k2 *ml = k1: m2 = k2***

**m3 *=* k3: m4 = k4 *m3* = *k3: m4* = *k4***

**nl = kl: n2 = k2 *n1 = k1: n2 = k2***

**n3 = k3: n4 = k4 *n3 = k3: n4 = k4***

**elseif d > mx then *инеc d > mx то***

**mx *=* d *mx* = *d***

**ml *=* kl: m2 = k2 *m1* = *k1: m2* = *k2***

**m3 = k3: m4 = k4 *m3= k3: m4 = k4***

**elseif d < mn then *инеc d < mn то***

**mn = d *mn* = *d***

**nl = kl: n2 = k2 *n1 = k1: n2 = k2***

**n3 = k3: n4 = k4 *n3* = *k3: n4* = *k4***

**end if *кесли***

**next 1 *кцикл***

**? «максимальный маршрут:» *вывод («максимальный маршрут:»)***

**? ml; m2; m3; m4 *вывод (m1; m2; m3; m4)***

**? «длина =»; mx *вывод («длина =»; mx)***

**? «минимальный маршрут:» *вывод («минимальный маршрут:»)***

**? nl; n2; n3; n4 *вывод (n1; n2; n3; n4)***

**? «длина =»; mn *вывод («длина =»; mn)***

**end *кон***

**vvdan: 'ввод данных *алг «ввод данных»***

**restore tchks *загрузка-точек***

**for k = 1 to n *от k = 1 до п***

**read x(k),y(k) *ввод x(k),y(k)***

**? x(k),y(k) *вывод x(k),y(k)***

**next k *кцикл***

**for k = 1 to n *от k = 1 до п***

**for l = 1 to n *от l = 1 до п***

**dx = x(k) - x(l) *dx* = *x(k) - x(l)***

**dy = y(k) - y(l) *dy* = *y(k) - y(l)***

**rs *=* dx\*dx + dy\*dy *rs = dx\*dx* + *dy\*dy***

**r(k,l) = sqr(rs) *r(k,l) = sqr(rs)***

**next 1 *кцикл***

**next k *кцикл***

**return *кон***

**mrshrt: 'маршруты:**

**data 1, 2, 3, 4**

**data 1, 2, 4, 3**

**data 1, 3, 2, 4**

**data 1, 2, 4, 3**

**data 1, 4, 2, 3**

**data 1, 4, 3, 2**

**tchks: 'координаты точек**

**data 0, 0**

**data 0, 3**

**data 4, 0**

**data 4, 3**

Результаты выполнения на ЭВМ приведенной программы:

**координаты точек:**

**0 0**

**03**

**4 0**

**4 3**

**маршруты: длина:**

**1 2 3 4 16**

**1 2 4 3 14**

**1 3 2 4 18**

**1 2 4 3 14**

**1 4 2 3 18**

**1 4 3 2 16**

**максимальный маршрут:**

**1 3 2 4**

**длина =18**

**минимальный маршрут:**

**1 2 4 3**

**длина = 14**

Четвертую задачу можно отнести к геометрическим задачам, ре­шение которых опирается на некоторые геометрические законы и свойства. Эта задача наиболее сложная среди рассмотренных задач из-за необходимости привлечения определенных математических знаний для организации ее решения.

**Задача 4.** «Ломаная».

Найти все точки самопересечения разноцветной замкнутой линии, заданной на плоскости координатами своих вершин в порядке обхода ломаной. Данные о ломаной представляются таблицей:

|  |  |
| --- | --- |
| **х** | **у** |
| 0 | 1 |
| 1 | 0 |
| 0 | 1 |
| 1 | 1 |

Особенность этой задачи - большое число частных случаев, свя­занных с возможным вырождением или наложением отрезков ло­манной линии. Именно эти ситуации и составляют содержание те­стов, на которых большинство программ дают неправильные резуль­таты.

Приведем проверочные тесты:

**Tecт1.** (Основной случай)

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

**Правильные результаты:**

точки пересечения

* 1. 0.5

**Тест** **2**. (Основной случай)

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 1 |
| 1 | 1 |
| 1 | 0 |

**Правильные результаты:**

точки пересечения:

отсутствуют

**Тест3.** (Наложение вершины)

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 1 |
| 0.5 | 0 |
| 1 | 1 |
| 1 | 0 |

**Правильные результаты:**

точки пересечения

* 1. 0

**Тест4.** (Наложение ребра)

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 1 |
| 0.2 | 0 |
| 0.8 | 0 |
| 1 | 1 |
| 1 | 0 |

**Правильные результаты:**

отрезок пересечения:

[0.2, 0] - [0.8, 0]

Для систематического конструирования алгоритмов и программы необходима разработка сценария диалога и описание метода решения поставленной геометрической задачи.

Сценарий

точек: <n>

координаты точек:

<k>: <x> <у>

……..

точки пересечения:

отрезок: <k> - <k+l> \*

отрезок: <1> - <1+1>

точка: <х> <у>

………

отсутствуют

Метод решения данной задачи может быть основан на вычислении точек пересечения отрезков (х1, у1) - (x2, у2) и (х3, y3) - (х4, y4) как точек пересечения линий, проходящих через заданные отрезки, с помощью системы уравнений:

(y2 – y1 )⋅( x – x1) - (x2 – x1)⋅(y – у1) = 0;

(у4 – у3)⋅(x – x3) - (x4 – x3)⋅(у – y3) = 0.

Решение этих уравнений может быть проведено вычислением определителей D, Dx, Dy приведенной системы уравнений:

(у2 – у1)⋅х - (х2 – х1)⋅у = (у2 – y1)⋅х1 - (x2 – x1)⋅y1;

(у4 – y3)⋅х - (х4 – х3) ⋅у = (у4 – у3)⋅х3- (x4 – x3)⋅y3,

для которой будет справедлив следующий набор расчетных формул:

х = Dx/D;

у = Dy/D;

D = (у2 **-** у1)⋅(х4 - x3) - (x2 - x1)⋅(y4 - y3);

Dx = [(y2 - yl)⋅xl - (х2 – x1)⋅y1] - (x4 – х3) - (x2 – x1)⋅[(y4 – y3)⋅x3 - (х4 – х3)⋅y3];

Dy = (у2 - у1)⋅[(у4 – у3)⋅х3 - (x4 - x3)⋅у3] - [(у2 – y1)⋅x1 - (х2 – x1)⋅y1]⋅(y4 – y3).

Факт пересечения пар отрезков может быть установлен из этих же уравнений подстановкой в правые части координат точек альтерна­тивного отрезка и сравнением значений этих выражений. А именно отрезок [(х3, у3) - (х4, у4)] пересекает линию, проходящую через отрезок [(x1, y1) - (х2, у2)], если эти выражения имеют разные знаки:

(у2 - у1)⋅(х3 – x1) - (х2 – х1)⋅(y3 – у1) × (у2 - у1)⋅(х4 – x1) - (х2 – x1)⋅(y4 – y1) ≤ 0.

Соответственно, отрезок [(х1, у1) - (х2, у2)] пересекает линию, проходящую через отрезок [(х3, у3) - (х4, у4)], если аналогичные выражения имеют разные знаки:

(у4 – у3)⋅(х1 – х3) - (х4 – х3)⋅(у1 – у3)×(у4 – у3)⋅(х2 – х3) - (х4 – х3)⋅(у2 – у3) ≤ 0.

И наконец, самый тонкий момент - это частные случаи, когда отрезки ломаной оказываются на одной прямой линии. В этом случае отрезки либо вообще не пересекаются, либо имеют общую часть, которую можно определить из взаимного расположения отрезков на прямой.

В последнем случае общая часть отрезков находится из взаимо­расположения отрезков [(х1, у1) - (х2, у2)] и [(х3, у3) - (х4, у4)] на прямой. В данной ситуации взаиморасположение вершин отрезков можно выяснить, вычислив взаиморасположение между ними на прямой относительно отрезка [(х1, у1) - (х2, у2)] по следующим фор­мулам:

d1 *=* 0*;*

d2 *=* (х2 – х1)⋅(х2 – х1) + (у2 – у1)⋅(у2 - 1);

d3 = (х3 – х1)⋅(x2 – х1) + (у3 - у1)⋅(у2 - 1);

d4 = (х4 – х1)⋅(х2 – х1) + (у4 – y1)⋅(y2 - 1).

Если d2 < min (d3, d4) или max (d3, d4) < 0, то отрезки не пересе­каются. В противном случае необходимо выделить и отбросить две крайние точки, и тогда оставшиеся две точки зададут общую часть этих отрезков.

Опираясь на эти **математические факты** можно приступить к составлению алгоритмов и программы. Приведем программу, в которой установлено максимальное число точек nt = 200. Реальное число точек устанавливается при вводе исходных данных из перечня операторов data, записанных в конце текста программы.

**′ самопересечение ломаной**

**nt = 200**

**dim x(nt), y(nt)**

**gosub wod 'ввод данных**

**? «точки пересечения:»**

**np = 0 'число пересечении**

**for k = 1 to nt - 1**

**xl = x(k): yl = y(k)**

**x2 = x(k + I): y2 *=* y(k + 1)**

**for 1 *=* k + 1 to nt - 1**

**x3 = x(I): y3 = y(I)**

**х4 = x(I + 1): y4 = y(I + 1)**

**gosub pint 'пересечение**

**next 1**

**next k**

**if np = 0 then ? «отсутствуют»**

**end**

**pint: ′ точка пересечения:**

**d213 = (у2 - yl)\*(x3 - х1) - (х2 - х1)\*(у3 - у1)**

**d214 = (у2 - у1)\*(х4 - х1) - (х2 - х1)\*(у4 - у1)**

**d431 *=* (у4 - у3)\*(х1 - хЗ) - (х4 - х3)\*(у1 - уЗ)**

**d432 = (у4 - у3)\*(х2 - хЗ) - (х4 - х3)\*(у2 - уЗ)**

**if d213\*d2l4 > 0 or d431\*d432 > 0 then**

**' нет пересечения**

**elseifd213\*d214 < 0 or d431\*d432 < 0 then**

**gosub tchki ' расчет точки**

**else ' отрезки на одной прямой**

**gosub lin 1**

**end if**

**return**

**tchki: ' расчет точки пересечения**

**np = np+1**

**? «отрезок:»; k; k + 1**

**? «отрезок:»; I; I + 1**

**D = (у2 - yl)\*(x4 - хЗ) - (х2 - х1)\*(у4 - уЗ)**

**Dx = [(у2 - у1)\*х1 - (х2 - х1)\*у1]\*(х4 - хЗ)**

**Dx = Dx - (х2 - х1)\*[(у4 - у3)\*х3 - (х4 - х3)\*у3]**

**Dy *=* (у2 - у1)\*[(у4 - у3)\*х3 - (х4 - х3)\*у3]**

**Dy *=* Dy - [(у2 - yl)\*xl - (х2 - х1)\*у1]\*(у4 - уЗ)**

**х *=* Dx/D**

**у *=* Dy/D**

**? х; у**

**return**

**lin 1: 'отрезки на одной прямой**

**d2 *=* (х2 - х1)\*(х2 - х1) + (у2 - у1)\*(у2 - 1)**

**d3 *=* (хЗ - х1)\*(х2 - х1) + (уЗ - у1)\*(у2 - 1)**

**d4 = (х4 - xl)\*(x2 - х1) + (у4 - у1)\*(у2 - 1)**

**if d3 > d2 and d4 > d2 then**

**' нет пересечения**

**Iseif d3 < 0 and d4 < 0 then**

**' нет пересечения**

**else ' отрезки пересекаются:**

**gosub otrеz ' общий отрезок**

**end if**

**return**

**otrez: 'расчет общего отрезка**

**np = np + 1**

**? «отрезок пересечения:»**

**if d3 < 0 or d4 < 0 then**

**? х1; у1; «-»**

**elseif d3 < d4 then**

**? х3; у3; «-»**

**else**

**? х4; у4; «-»**

**end if**

**if d2 < d3 or d2 < d4 then**

**? х2; у2**

**elseif d3 < d4 then**

**? x3; y3**

**else**

**? х4; у4**

**end if**

**return**

**vvod: ' ввод данных**

**restore test1**

**read n**

**? «точек:»;nt**

**for k = 1 to nt**

**read x(k), y(k)**

**? x(k); y(k)**

**next kn**

**t = nt + 1**

**x(nt) = x(l)**

**y(nt) = y(l)**

**return**

**test1: 'точки ломаной:**

**data 4**

**data 0, 0**

**data 1, 0**

**data 0, 1**

**data 1, 1**

**test2: 'точки ломаной:**

**data 4**

**data 0, 0**

**data 1, 0**

**data 0, 1**

**data 1, 1**

В тексте данной программы записаны два варианта тестовых данных, смена которых может быть проведена изменением имени метки test1 или test2 в операторе перезагрузки restore в подпрограмме ввода данных.

## 6.5. Технология дистанционного обучения

Дистанционное образование - это новая технология обучения, основанная на использовании персональных компьютеров, электрон­ных учебников и сетей телекоммуникации. Эта новая технология и форма обучения самым тесным образом связана с развитием сети Интернет [7, 8, 9].

В **Российской Федерации** подготовлен законопроект, по которому дистанционное образование приравнивается к традиционным фор­мам очного, заочного и вечернего обучения. В настоящее время пять ведущих вузов России ведут эксперименты по отработке технологий дистанционого обучения студентов.

В Республике Казахстан дистанционные формы обучения полу­чили официальное признание после принятия нового закона об об­разовании летом 1999 года. В Законе об образовании Казахстана дистанционное обучение определено как «одна из форм обучения лиц, находящихся в отдалении от организаций образования, с помощью электронных и телекоммуникационных средств».

Началом распространения новых компьютерных технологий обу­чения послужило введение в середине 80-х годов курса информатики во всех средних школах нашей страны [16]. Основной целью школь­ного курса информатики с конца 80-х годов было обучение всех учащихся компьютерной грамотности - умениям читать, писать и получать информацию с помощью персональных ЭВМ [17].

Эксперименты с дистанционным образованием в вузах России начались с середины 90-х годов. **Технологической базой** для дистан­ционного обучения являются персональные компьютеры, электронные учебники и вычислительные сети. Развитие сети телекоммуникаций и появление образовательных серверов в сети Интернет сделало реальностью распространение новых технологий дистанционного обучения [7, 8. 9].

За рубежом развитие сети Интернет в 90-х годах привело к появ­лению первых электронных университетов как новых компьютер­ных форм получения образования. Насыщение персональными компьютерами университетов, колледжей и средних школ создает предпосылки для создания новых дистанционных форм обучения студентов и старшеклассников у нас в стране и зарубежом.

В США, Великобритании, Австралии, Канаде, Германии развитие сети Интернет создало условия для организации сетей дистанцион­ного обучения, переживающих настоящий бум. При этом наиболее продвинутые проекты дистанционного образования развиваются на базе или при поддержке крупнейших компьютерных фирм - IBM, Apple, DEC, Sun, Novel, Microsoft и т. д.

Одним из наиболее известных зарубежных проектов дистанцион­ного обучения является создание **Открытого университета Велико­британии.** В этом университете обучение в форме компьютерных телеконференций началось в конце 80-х годов.

Основной **технологической идеей** этого проекта была передача заданий и результатов их выполнения с помощью электронной почты и обсуждение работ посредством телеконференций. К середине 90-х годов эти курсы пользовались успехом более чем у 5000 студен­тов, имевших домашние компьютеры.

Однако главным достижением проекта стали **комплекты бумажных учебников,** изданные университетом для дистанционного обучения и подготовки менеджеров в области информационных технологий. Это позволило Открытому университету перейти к дистанционному обучению студентов на международном уровне.

Одна из ветвей этого международного проекта получила развитие в России на базе образовательной **корпорации LINK,** использующей эти учебники. В настоящее время в Открытом Университете в дис­танционной форме обучается несколько тысяч студентов.

Вторым по развитию у нас в стране является проект **Института дистанционного образования** Московского государственного уни­верситета экономики, статистики и информатики (МЭСИ). В рос­сийском проекте дистанционного образования в настоящее время обучается более 25 тысяч студентов по различным экономическим специальностям, для которых созданы соответствующие комплекты бумажных и электронных учебников [6, 7].

Еще один проект дистанционного образования был развернут в середине 90-х годов **Российским Государственным технологическим** **университетом** (МАТИ) на базе более 120 школ Москвы, Подмос­ковья, Челябинска и Приднестровья. Целью этого проекта была дистанционной подготовка учащихся средних школ к вступитель­ным экзаменам в вузы Российской Федерации.

Для данного проекта были созданы учебные пособия для посту­пающих в вузы по **информатике, математике** и **английскому языку** с использованием тестов, электронных учебников и электронных за­дачников. В рамках данного проекта был разработан и апробирован электронный учебник по информатике, используемый в настоящем учебном пособии [20].

Особую роль в развитии дистанционного образования, на наш взгляд, может и должен сыграть **вузовский курс информатики,** который позволит отработать технологию компьютерного обучения с исполь­зованием сети Интернет и электронных учебников и организацией компьютерной технологии приема и сдачи экзаменов.

Новыми в дистанционном обучении для отечественной системы образования являются **электронные учебники** с использованием компьютерных методов тестирования знаний. Отличительной осо­бенностью электронных учебников является то, что работа с ними имеет форму диалога, в ходе которого учащиеся усваивают знания в виде фактов, вопросов, суждений, утверждений и т. п.

Кроме того, в электронных учебниках система контрольных тестов позволяет учащимся **самостоятельно проверять** усваиваемые знания. Большую помощь при этом могут оказать бумажные учеб­ники, поскольку в них могут быть найдены ответы на вопросы, которые заложены в электронный учебник.

В этом назначении электронные учебники для систем дистанци­онного обучения могут выполнять функции репетиторов, оказыва­ющих учащимся помощь в изучении различных учебных дисциплин. Консультации преподавателей при этом переносятся на обсуждение индивидуальных заданий с учетом интересов учащихся.

Настоящий **бумажный учебник** по информатике вместе со своими электронными версиями - пример новой технологии обучения, которая может использоваться в вузах и школах для изучения курса информатики, в домашних условиях и учебных центрах - для само­образования и дистанционного обучения.

**Необходимым условием** для этого является возможность доступа к персональным компьютерам и электронной версии настоящего учеб­ника. Для работы в системе дистанционного обучения необходим доступ к сети Интернет, а также электронный почтовый ящик, по которому можно получать электронную корреспонденцию из центра дистанционного обучения.

**Сетевые электронные версии** настоящего учебника по информати­ке можно найти в сети Интернет с помощью информационно-поис­ковых систем Яндекс или Апорт по запросу «Каймин учебник ин­форматика». Отличием этих электронных учебников является то, что в них можно не только листать и читать текст учебника, но и найти тесты, с помощью которых можно перепроверить свои знания по информатике.

Данные тесты по информатике с 1997 года используются в систе­ме **дистанционного контроля знаний** учащихся средних школ, создан­ных на психологическом факультете Московского Государственного Университета и получившей название Телетестинг. Использование психологами этих тестов в течение последних трех лет показало уди­вительные результаты.

Все школьники, прошедшие **телетестинг** (дистанционный конт­роль знаний), показали самый высокий уровень компьютерной грамотности. Однако на вопросы об алгоритмизации и о програм­мировании выпускники школ в основном продемонстрировали удов­летворительные либо даже неудовлетворительные знания.

Объяснением этим фактам может служить следующее. Участ­вовать в телетестинге могут только те школьники, которые **умеют работать на ЭВМ,** а для этого необходимо получить соответствую­щие знания. Самые первые учебники ориентировались на изучение основ алгоритмизации без работы и отладки программ на ЭВМ [16].

В **новейших учебниках** по информатике изучение элементов про­граммирования либо вовсе отсутствует, либо проходит в отрыве от изучения основ алгоритмизации [18, 19, 21]. По этой причине знания учащихся либо фрагментарны, либо неверны.

Освоению компьютерной грамотности с последующим изучением алгоритмизации и отладки программ на ЭВМ был посвящен **базовый школьный учебник** [17], признанный лучшим в 1997 году. Однако он не переиздавался и стал недоступен.

Настоящий учебник, созданный в соответствии с **требованиями государственных стандартов** образования, ориентирован на изучение студентами современного уровня развития информатики как инфор­мационной индустрии и освоение техники решения профессиональ­ных задач на ЭВМ.

В этом смысле он является **развитием и продолжением** нашего школьного учебника по информатике и построен так, чтобы его можно было использовать как в условиях очного, так и дистанцион­ного обучения. При этом в учебник введена линия на **опережающее изучение Интернет,** для того чтобы студенты как можно скорее могли получить доступ к его информационным ресурсам, электронным книгам, журналам и центрам дистанционного образований.

Самую большую **библиотеку электронных книг и учебников** по экономике можно найти на серверах Института дистанционного образования МЭСИ. В этой библиотеке можно найти электронную версию настоящего учебника по информатике для студентов.

В настоящее время практически **все ведущие вузы Российской Федерации,** имеющие отделения или факультеты информатики или информационных технологий, включились в развитие технологий дистанционного обучения и ввели в той или иной форме вступи­тельные экзамены или испытания по информатике.

**Подробную информацию о всех Российских вузах** и новостях дис­танционного образования можно найти в сети Интернет на сервере Информатика Министерства образования Российской Федерации.

# Приложение

# ИНТЕРПРЕТАТОР ЯЗЫКА ПРОЛОГ

## 1. Назначение интерпретатора Пролога

Интерпретатор языка Пролог предназначен для проведения прак­тикума на персональных компьютерах с базами знаний, экспертными системами и изучением принципов логического вывода в системах искусственного интеллекта.

Данный интерпретатор может использоваться на персональных компьютерах IBM PC с операционной системой MS DOS или Windows. Для работы интерпретатора достаточно иметь оперативную память не менее 250 Кбайт и накопитель на гибком или жестком диске.

## 2. Запуск интерпретатора Пролога

Интерпретатор Пролога находится на жестком диске или дискете в каталоге с именем PROLOG. Запуск интерпретатора проводится с помощью команды

**> prolog**

из каталога, в котором он находится.

Интерпретатор может быть запущен одновременной с загрузкой некоторой базы знаний или программы на Прологе следующей ко­мандой:

**> prolog <имя>**

где **<имя>** - это имя некоторого файла данных. Если запуск прошел успешно, то на экране появится изображение такого вида:

**(С) Файл Диалог Окна Свободно: 303Кб.**

**18:40:54**

**- [\_ ] --------------- D:\PROLOG\FAMILY.PRL ——————1-[208]-**

**мама (зина, вова);**

**папа (вова, лена);**

**бабушка (х, z) <- мама (х, у), мама (у, z);**

**бабушка (х, z) <- мама (х, у), папа (у, z);**

**F1 Подсказка F3 Открыть Alt-F3 Закрыть F4 Диалог F10 Меню**

Настоящая версия диалогового интерпретатора языка Пролог предоставляет следующие возможности:

* работа в диалоге с базами данных и программами на Прологе;
* поиск и чтение программ и баз данных, записанных на дисках;
* ввод и редактирование новых программ и баз данных;
* запись на диски новых или отредактированных программ;
* вывод на печать программ и баз данных на Прологе.

Для работы с интерпретатором используются следующие клави­ши-команды:

**F1** - получение подсказки;

**F2** - работа с файлами;

**F3** - открытие нового окна;

**Alt-F3** - закрытие текущего окна;

**F4** - начало диалога с программой;

**F10** - обращение к верхнему меню.

## 3. Диалог с программами на Прологе

**Диалог** с программами на Прологе начинается после нажатия клавиши **F4** - начало диалога с программой.

Результатом выполнения этой команды будет появление на экра­не окна диалога:

**Файл Редактор Диалог**

**-РЕДАКТОР –**

**мама ( ДИАЛОГ**

**папа ( ? мама(х,вова);**

**бабуш х = зина**

**бабуш ДРУГИХ РЕШЕНИЙ НЕТ**

**F1 Подсказка F3 Открыть Alt-F3 Закрыть F4 Диалог**

В режиме «**Диалог**» можно вводить вопросы по отношению к фактам и правилам, имеющимся в программе или базе данных, которые размещены в оперативной памяти ЭВМ. Поиск ответов на вопросы начинается нажатием клавиши ввода **Enter.** Ответы на вопросы выводятся здесь же в окне диалога вслед за вопросом.

В окне диалога можно, задать **серию вопросов** к текущей программе или базе знаний, вернуться к любому вопросу и запустить его еще раз с новыми данными, а также просматривать протокол диалога в любом порядке.

Для **завершения диалога** с программой необходимо закрыть окно диалога с помощью команды **Alt-F3** - закрытие текущего окна.

## 4. Ввод и редактирование программ

Для **ввода** новых программ или редактирования имеющихся не­обходимо перейти в режим «Редактор». Для перехода к вводу новой программы или базы данных необходимо нажать клавишу F3 - от­крытие нового окна - и приступить к построчному вводу текста программы.

Ввод и **редактирование** текстов программ и баз данных на Проло­ге выполняется по правилам, традиционным для редакторов текс­тов.

Признаком конца строки служит нажатие клавиши ввода **Enter.** Последний введенный символ стирается нажатием клавиши **BS**. Удаление символов в текущей позиции проводится нажатием кла­виши **Del.**

Для **исправления** текущей строки используются клавиши управле­ния курсором - стрелки вправо-влево, а также клавиши перехода в начало строки **Ноmе** и перехода в конец строки **End.**

Для **вставки** символов необходимо перейти в режим «Вставка», нажав клавишу **Ins.** Признаком режима «Вставки» служит размер курсора: в режиме «Вставка» видна половина курсора, а в режиме «Замена» курсор целиком закрывает текущий символ. Переключе­ние режимов вставка/замена производится нажатием клавиши **Ins.**

**Рассечение** строки осуществляется в режиме «Вставка» нажатием клавиши ввода **Enter.** Для обратной операции - склейки строк кур­сор переводится в конец верхней строки, где нажимается клавиша удаления символов **Del.** В результате нижняя строка «приклеится» к верхней. Удаление текущей строки произойдет при одновременном нажатии клавиш **Ctrl и Y.**

Для **удаления** всего **текста** в целом для начала ввода нового текста программы необходимо перейти в режим «Файл», нажав клавишу **F2,** а затем указать в появившемся подменю режим «Новый».

## 5. Операции с файлами

Для выполнения операций с файлами на дисках - записи, чте­ния, поиске, смене каталогов в настоящем интерпретаторе необхо­димо перейти в режим «Файл» нажатием клавиши **F2** - работа с файлами. Результатом будет появление на экране следующего меню:

**Открыть.. F3**

**Сохранить F2**

**Сохранить как..**

**Смена Каталога..**

**Вызов DOS**

**Выход Alt-X**

При переходе в режим «Открытие файлов» на экране появится окно:

**-[\_] — Открыть окно —**

**Имя**

\*.\*

**Файлы**

**APP.PRL PROLOG. EXE Открыть**

**BLOK1.PRL PROLOG.HLP Отмена**

**FAMILY.PRL ..\ Помощь**

**HOM1.PRL**

**HOMES.PRL**

**D:\PROLOG\\*.\***

**APP.PRL 0 Сен 8, 1991 5:01pm**

В этом окне перечислены имена файлов в текущем каталоге, ука­занном в предпоследней строке (обычно в каталоге PROLOG).

Для **выбора файла** из текущего каталога необходимо нажать кла­вишу TAB, затем стрелками вверх и вниз указать имя нужного файла и нажать клавишу ввода Enter.

Для **перехода** в другой **подкаталог,** имя которого заканчивается символом « \ », требуется выбрать это имя и нажать клавишу ввода **Enter.** Для возврата в предыдущий каталог выбирается имя « ../ ».

Для **записи** текстов программ на диски необходимо перейти в режим работы с файлами (**F2** - работа с файлами) и выбрать один из двух режимов:

**Сохранить F2**

**Сохранить как ...**

Режим «Сохранить как ...» из меню «Файл» служит для сохране­ния файла, открытого в текущем окне, под другим именем. При выполнении этой операции на экране появится следующее окно, в котором нужно будет указать новое имя файла:

**[\_] ----- Сохранить как ----**

**Новое имя: D:\PROLOG\HOMES.PRL**

**Окей Отмена**

При указании в меню «Работы с файлами» режима «Смена ката­лога» на экране ЭВМ появится панель «Смена каталога», в которой указывается текущий каталог или устройство.

Для **завершения работы** с интерпретатором необходимо перейти в меню «Файл» и указать в нем подпункт «Выход». Этого же можно достичь нажатием клавиш **Alt-X.**

Если в каком-либо окне осталась измененная, но несохраненная программа или база данных, то вы будете предупреждены об этом.

## 6. Краткое описание языка Пролог

### 6.1. Базовые понятия языка Пролог

Основными понятиями в языке Пролог являются факты, правила и вопросы. Из фактов и правил образуются описания данных, про­цедур и программы на языке Пролог. Вопросы - это основное сред­ство ведения диалога с базами знаний и программами, имеющимися в памяти ЭВМ.

Факты в Прологе служат для описания конкретных данных и простейших сведений. Примеры фактов:

**мама (зина, вова);** - Зина - мама Вовы

**папа (миша, вова);** - Миша - папа Вовы

Группы фактов могут образовывать данные. Совокупность дан­ных, размещаемых на дисках, образуют базы данных. Общее опреде­ление данных в Прологе:

данные:

**<факт>; [<факт>;...]**

Правила используются для описания определений, процедур принятия решений и обработки данных. Примеры использования правил для описания определения понятия «родитель»:

**родитель (х,у) ← папа (х,у);** - Родитель — это папа или мама

**родитель (х,у) ← мама (х,у);**

**Процедуры** образуются из некоторых групп правил. Общая форма описания процедур в Прологе:

процедура:

**[<факты>] <правило>; [<правило>; ...]**

Пример описания рекурсивной процедуры, в которой определя­емое понятие задается через самое себя:

**предок (х,у) ← родитель (х,у);**

**предок (x,z) ← родитель (х,у), предок (y,z);**

Программа на Прологе — это совокупность процедур над опреде­ленными данными:

программа:

**<процедуры>; [<данные>;]**

**Описания баз данных** на Прологе образуют совокупность описа­ний данных:

база данных:

**<данные>; [<данные>; ... ]**

Базы знаний на Прологе описываются наборами фактов и правил определения обобщенных понятий над ними:

база знаний:

**<данные>; <правила>;**

Вопросы в языке Пролог служат запросами к базам данных и знаний, а также обращениями к процедурам и программам. Приме­ры вопросов:

**? мама (х,вова) ?**

**предок (х,вова)**

### 6.2. Описания фактов

Факты в языке Пролог описываются в следующей форме:

факт:

**<имя>(<арг>[,<арг> ...]);**

Описание факта начинается с имени. В роли имен могут исполь­зоваться любые последовательности букв и цифр, начинающиеся с буквы;

имя:

**<буква>[<буква><цифра> ...]**

Буквы могут быть выбраны из русского и латинского алфавитов.

После имени в описаниях фактов задается список аргументов, заключаемых в круглые скобки. В конце описания каждого из фак­тов ставится точка с запятой.

В роли аргументов могут быть указаны любые слова, словосоче­тания и числа. Словосочетание - это любая последовательность слов, соединенных знаком подчеркивания.

Словосочетание:

**<слово>[<слово> ...]**

Слова, как и имена, - это любые последовательности из букв и цифр, начинающиеся с букв:

слово:

**<буква>[<буква><цифра> ...]**

Числа в данной реализации Пролога - это только целые числа (отрицательные - со знаком минус):

число:

**[—]<цифра>[<цифра> ...]**

Примеры записи чисел - 0, 1, +3, -25.

С е м а н т и к а : каждый факт в Прологе интерпретируется как некоторое истинное утверждение. Опираясь на эти утверждения, интерпретатор Пролога рассматривает различные вопросы как утверждения, которые должны быть доказаны или опровергнуты исходя из имеющихся фактов и правил.

### 6.3. Правила записи вопросов

**Вопросы в Прологе** служат для записи простых или сложносоставных запросов к базам знаний или обращений к процедурам и про­граммам. Ответами на запросы к базам данных и знаний могут быть логические значения ДА (истина) или НЕТ (ложь) либо список кон­кретных данных, отвечающих запросу.

Общая форма записи сложносоставных вопросов:

сложный вопрос:

**? <вопрос>[,<вопрос> ...]**

В сложносоставных вопросах составляющие его подвопросы раз­деляются запятыми. Любой вопрос в Прологе начинается со знака вопроса «?» и заканчивается знаком «точка с запятой». Пример сложносоставного вопроса:

**? папа** **(х,у), папа (у,вова);**

Простые вопросы имеют следующую форму записи:

простой вопрос:

**? <имя>(<арг>[,<арг> ...])**

Здесь **<имя>** - это имя некоторого факта или правила в програм­ме или базе знаний. Количество аргументов в таком простейшем вопросе должно строго совпадать с количеством аргументов (пара­метров) в описаниях соответствующих фактов и правил.

В роли аргументов в вопросах могут указываться как конкретные значения, так и переменные. Если в вопросе указаны только кон­кретные значения, то ответом будет логическое значение **ДА** или **НЕТ.** Если же в вопросе указываются переменные, то при положительном ответе дополнительным результатом будет вывод конкретных значе­ний переменных, указанных в исходном вопросе.

В роли переменных в Прологе могут указываться латинские и русские буквы, как строчные, так и прописные: **а, b,** **с,** ..., **х,** **у, z;** **A**, **В, С, ..., X, Y, Z;** **а, б, в, ...** **, э, ю, я; А, Б, В, ..., Э, Ю, Я.**

Кроме того, в качестве имен переменных могут употребляться лю­бые слова и словосочетания, заканчивающиеся знаком апострофа ['].

Примеры имен: **дед '**, **х1** **'**, **оценка4 '** и т. п.

Кроме того, в вопросах и правилах на Пролог аргументы могут иметь неопределенные значения. Для этого вместо конкретных значений или имен переменных в вопросе указывается знак подчер­кивания [ \_ ].

Пример вопроса с использованием неопределенных значений:

**? мама (мать', \_ );**

Ответом на указанный вопрос будет перечень имен всех матерей, сведения о которых имеются в базе знаний.

### 6.4. Запись правил и процедур

Правила в Прологе используются для описания определений, запросов к базам данных, а также обращений к другим правилам и процедурам. Примеры записи правил:

**родитель (х,у) ← мама (х,у);**

**родитель (х,у) ← папа (х,у);**

**бабушка (x,z) ← мама (х,у), родитель (y,z);**

**дедушка (x,z) ← папа (х,у), родитель (y,z);**

Общая форма записи правил на Прологе:

**правило:**

**<заголовок> ( <определение>;**

Здесь <заголовок> отделяется от <определения> двумя знаками: ← «меньше» и «тире». В конце правила записывается «точка с запя­той».

Заголовки правил имеют следующую форму:

**заголовок:**

**<имя>(<парам>[,<парам> ...])**

Здесь <имя> - это любое слово или словосочетание. В роли параметров в заголовках правил могут указываться переменные или конкретные значения либо неопределенные значения с помощью знака подчеркивания [\_].

Описание заголовка служит образцом записи обращений к правилам и процедурам. Определения в правилах образуются из обращений к другим правилам и процедурам, а также из запросов к данным и встроенным процедурам Пролога. Общая форма записи определений в правилах:

**определение:**

**<заголовок>[,<заголовок> ...]**

Запросы и определения отделяются в описаниях правил запятыми и записываются по тем же правилам, что и заголовки.

Порядок записи обращений и запросов в определениях правил в языке Пролог весьма существенен, поскольку он задает порядок их применения и выполнения. Нарушение этого порядка может при­вести к значительному изменению конечных результатов выполнения правил и процедур.

Результатом выполнения любых процедур и правил в Прологе всегда являются логические значения - **ДА** либо **НЕТ,** соответству­ющее успеху или неуспеху их применения. В этом смысле приме­нение правил в Прологе совпадает с проверкой условий в логике. Соответственно форма записи обращений к правилам и процедурам в Прологе совпадает с формой записи предикатов в математической логике.

Однако в отличие от математической логики, где существенны только логические значения этих предикатов, в Прологе резуль­татом применения правил и процедур может быть конкретизация значений переменных, указанных в обращениях к ним.

Пример такого обращения:

**? родитель (х,вова)**

**х = зина**

Полученный ответ имеет двойной результат: во-первых, подтверж­дение истинности наличия «**родителей**» у «**вовы**», а во-вторых, кон­кретизация переменной «**х**» в обращении к процедуре «**родитель**» с именем «**вова**», которая дает значение «**х = зина**».

Кроме конкретизации значений переменных в запросах и обра­щениях результатами выполнения процедур и правил в Прологе могут быть различного рода побочные эффекты - вывод на экран линий, точек, картинок, сообщений, запись и считывание данных и т. п.

В этом смысле результаты выполнения процедур и программ на Прологе могут быть эквивалентны результатам выполнения программ на других языках программирования.

### 6.5. Встроенные предикаты Пролога

Для операций над числами в рассматриваемой версии Пролога имеются следующие арифметические предикаты:

Предикат: Смысл:

**СУММА (x,y,z)** х + у = z

**ПРОИЗВ (x,y,z)** х\*у = z

**ЧАСТНОЕ (x,y,z)** [х/у] = z

**ОСТАТОК (x,y,z)** х - [х/у] = z

**РАВНО (х,у)** х = у

**БОЛЬШЕ (х,у)** х > у

**МЕНЬШЕ (х,у)** х < у

Арифметические предикаты по смыслу и функциям существенно отличаются от арифметических операций в традиционных языках программирования. Прежде всего любой из этих предикатов выра­батывает логическое значение **ИСТИНА** или **ЛОЖЬ** в зависимости от того, имеет или не имеет решение арифметическая задача, пред­ставляемая предикатом.

Примеры:

**? СУММА (2,2,5)**

**НЕТ**

**? СУММА (2,2,z)**

**z = 4**

**? СУММА (х,2,5)**

**х = 3**

Таким образом, арифметические предикаты позволяют решать простейшие арифметические уравнения с одной или несколькими переменными, автоматически производя соответствующие вычисле­ния, если эти решения существуют. Однако обращение к предика­там, связанным с умножением и делением целых чисел, может дать отрицательный результат из-за отсутствия решений в некоторых случаях:

**? ПРОИЗВ (2,у,5)**

**НЕТ**

**? ЧАСТНОЕ (5,2,z)**

**z = 2**

**? ОСТАТОК (5,2,r)**

**r = 1**

Предикаты БОЛЬШЕ и МЕНЬШЕ вместе с предикатом РАВНО могут использоваться для сравнения различных численных перемен­ных и констант. А предикаты «неменьше» и «небольше» могут быть определены через отрицание предикатов **БОЛЬШЕ** и **МЕНЬШЕ**:

**небольше (х,у) ← НЕ (БОЛЬШЕ (х,у));**

**неменьше (х,у) ← НЕ (МЕНЬШЕ (х,у));**

Предикат **РАВНО** может использоваться также для сравнения значений переменных и констант. При этом переменные могут быть и числовыми, и символьными, и других типов. Примеры:

**? РАВНО (5,z)**

**z= 5**

**? РАВНО (х,миша)**

**х = миша**

**? РАВНО (коля.оля)**

**НЕТ**

Вывод графических изображений на экран в настоящей версии интерпретатора может производиться с помощью следующих преди­катов:

**ТОЧКА** **(х,у,с);**

**ЛИНИЯ (x,y,u,v,c);**

**ОКРУЖНОСТЬ** **(х,у,с);**

где (х,у) и (u,v) - координаты точек на экране, а с - цвет точки. Конкретные значения кодов цветов и диапазоны значений коорди­нат точек на экране зависят от типа ЭВМ.

А. **Графические предикаты:**

Предикат: Смысл: Пример:

**ФОН (с)** задание цвета фона **ФОН (8)**

**ТОЧКА (х,у,с)** вывод точки **ТОЧКА (0,0,2)**

**ЛИНИЯ (x,y,u,v,c)** вывод отрезка линии **ЛИНИЯ (0,0,50,50,4)**

**ОКРУЖНОСТЬ (х,у,с)** вывод окружности **ОКРУЖНОСТЬ (90,90,10, 6)**

**ЗАКРАСКА** **(х,у,с)** закраска фигур **ЗАКРАСКА (10,40,8)**

Б. **Предикаты ввода-вывода:**

Предикат: Смысл: Пример:

**ВЫВОД (<3начение>)** вывод набора значений **ВЫВОД** **(«имя», х)**

**ВВОДСИМВ** (s) ввод символьных данных

**ВВОДЦЕЛ (n)** ввод числовых данных

**ПС** перевод строки

В. Логические предикаты:

Предикат: Смысл: Пример:

**ИСТИНА** логическая константа

**ЛОЖЬ** логическая константа

**НЕ (<Предикат>)** отрицание предиката **НЕ (РАВНО** **(z,0))**

# Толковый словарь

**Автор** - лицо, творческим трудом которого создано произведение (программа или база данных на ЭВМ).

**Авторские права** - права на имя, внесение изменений и имущест­венные права на созданное произведение.

**Алгоритм** - совокупность действий со строго определенными пра­вилами выполнения.

**Алгоритм массовый,** если он применим для решения некоторого класса задач.

**Алгоритм однозначный,** если однозначны правила его выполнения.

**Алгоритм правильный,** если он дает правильные результаты для любых допустимых данных и ситуаций.

**Алгоритм результативен,** если его выполнение завершается опреде­ленными результатами.

**Алгоритм содержит ошибки,** если его выполнение приведет к непра­вильным результатам, либо не дает никаких результатов.

**Алгоритмизация** - составление алгоритмов для решения поставлен­ных задач.

**Алгоритмические ошибки** - ошибки в методе, постановке, сценарии и реализации.

**Алгоритмические языки** - языки описания алгоритмов.

**Антивирусные программы** - программы для защиты от вирусов.

**База данных** - форма представления и организации совокупности данных, допускающие поиск и обработку с помощью ЭВМ.

**База знаний** - форма представления и организация совокупности данных и правил вывода, допускающих логический вывод и ос­мысленную обработку данных в ЭВМ.

**Байт** - 8-разрядное двоичное число (1 байт = 8 бит).

**Бит** - величина, принимающая значение 0 или 1.

**Безошибочное программирование** - составление алгоритмов и про­грамм без ошибок.

**Бейсик** - диалоговый язык программирования для персональных компьютеров.

**Бумажные технологии** - информационные технологии, связанные с бумагой.

**Вопрос в Прологе** - запрос к базе знаний или программе.

**Входные данные** - данные, вводимые в ЭВМ.

**Выходные данные** - данные, выводимые из ЭВМ.

**Вычислительная машина** - устройство для организации автоматичес­ких вычислений и обработки информации.

**Вычислительные процессы** - процессы вычислений и обработки информации в вычислительных машинах и системах.

**Вычислительная система** - объединение вычислительных машин для организации хранения, обработки и поиска информации.

**Вычислительная сеть** - совокупность ЭВМ, объединенных сетью связи, позволяющей компьютерам проводить обмен информацией.

**Вычислительные науки** - науки об организации вычислительных процессов, а также вычислительных машин, систем и сетей.

**Глобальная сеть** - сеть, в которой объединены компьютеры в раз­личных странах на различных континентах.

**Дизъюнкция** - логическая связка «или».

**Дизъюнкция** А или В **истинна,** когда истинно хотя бы одно из сужде­ний А или В.

**Дизъюнкция** А или В **ложна,** когда ложны оба суждения А и В.

**Дисплей** - устройство отображения информации на электронном экране.

**Дистанционное обучение** - обучение на расстоянии с использованием учебников, персональных компьютеров и сетей ЭВМ.

**Доказательство** - последовательность суждений, обосновывающих некоторое утверждение.

**Доказательное программирование** - составление алгоритмов и про­грамм одновременно с доказательством их правильности.

**Документ** - информация, зафиксированная на материальном носи­теле, имеющем реквизиты, позволяющие его идентифицировать.

**Закон достаточных оснований:** «Всякое утверждение должно предпо­лагать наличие аргументов и фактов, достаточных для его обосно­вания».

**Закон двойного отрицания;** «Если неверно отрицание, то суждение - истинно».

**Закон исключения третьего:** «Истинно суждение либо его отрицание - третьего не дано».

**Закон противоречия:** «Не могут быть истинны одновременно сужде­ние и его отрицание».

**Закон тождества:** «Предмет рассмотрения должен быть строго опре­делен и не должен меняться до конца обсуждения».

**Законы логики** - общие законы логического вывода.

**Защита информации** - действия и средства по предотвращению утечки, хищению, искажению или подделки информации.

**Импликация** - логическое следование А → В.

**Импликация** А → В **ложна,** когда посылка А истинна, а следствие В ложно.

**Импликация** А → В **истинна,** когда истинно следствие, либо ложны и следствие, и посылка.

**Имущественные права** - права собственности на произведение (про­граммы и базы данных для ЭВМ).

**Индукция** - вывод общих суждений из частных примеров.

**Интеллектуальная собственность -** результаты и продукты творчес­кого интеллектуального труда.

**Интеллект** - способность производить логические умозаключения и осмысленную переработку информации.

**Интернет** - международная компьютерная телекоммуникационная сеть.

**Интерпретатор** - это система интерпретации (выполнения) программ на ЭВМ.

**Информатика** - научная дисциплина, изучающая законы и методы накопления, обработки и передачи информации с помощью ЭВМ.

**Информация** - сведения о лицах, предметах, фактах, событиях, и процессах независимо от формы их представления.

**Информационная культура** - умение читать книги, искать данные, осмысливать информацию и предоставлять ее людям.

**Информационные процессы** - процессы сбора, обработки, накопле­ния, хранения, поиска и распространения информации.

**Информационные ресурсы** - совокупность документов в архивах, библиотеках, фондах, банках данных и других информационных системах.

**Информационная система** - организация совокупности документов с использованием средств вычислительной техники и телекомму­никационной связи.

**Информационные технологии** - технологии передачи, накопления и обработки информации.

**Информационные услуги** - действия людей, организаций и служб по сбору, поиску, хранению и предоставлению информации.

**Искусственный интеллект** - компьютерные системы, моделирующие или воспроизводящие интеллектуальную деятельность.

**Исходные данные** - данные, исходные по постановке решаемых задач.

**Калькуляция** - таблица с определенным набором надписей, формул и данных, используемых для расчетов.

**Компакт-диск** - оптический диск, используемый для постоянного хранения больших объемов информации.

**Компилятор** - система трансляции программ с одного языка на другой.

**Компьютер** - электронная вычислительная машина.

**Компьютерная грамотность** - умение читать, писать и искать инфор­мацию с помощью ЭВМ.

**Компьютерные вирусы -** саморазмножающиеся программы, способ­ные разрушать информацию на ЭВМ.

**Контрпример** - пример, опровергающий некоторое общее утверж­дение.

**Конфиденциальная информация** - информация, доступ к которой ограничен.

**Конъюнкция** - логическая связка «и».

**Конъюнкция** А и В **истинна,** когда истинны оба суждения А и В.

**Конъюнкция** А и В **ложна,** когда ложны суждения А или В.

**Курсор** - значок для указания места на экране ЭВМ.

**Лексикон** - редактор текстов для компьютеров IBM PC с операци­онной системой MS DOS.

**Лемма** - вспомогательное утверждение, предполагающее самостоя­тельное доказательство.

**Лицензия** - документ, подтверждающий права на использование программ для ЭВМ.

**Логика** - наука, изучающая правильность суждений, рассуждений и доказательств.

**Логические функции** - функции, принимающие логические значе­ния «истина» или «ложь».

**Логические элементы** - устройства, на входах и выходах, на которые могут появляться сигналы 0 или 1.

**Локальная вычислительная сеть** - сеть, объединяющая компьютеры в комнате или соседних помещениях.

**Магнитные диски** - средство хранения информации, программ и данных в ЭВМ.

**Массив** - область машинной памяти, выделяемая для размещения совокупности данных.

**Математическая индукция** - принцип доказательства истинности последовательностей утверждений.

**Математическая логика** - дисциплина, изучающая технику матема­тических доказательств.

**Массовая информация** - информация, предназначенная для неогра­ниченного круга лиц.

**Машинная программа** - последовательность машинных команд и данных, реализующих определенные функции ЭВМ.

**Метод** - единый способ решения некоторого класса задач.

**Метод правильный,** если он дает правильные результаты для любой задачи данного класса.

**Метод неправильный,** если существует задача данного класса, для которой он дает неправильные результаты или вовсе не дает результатов.

**Мобильность программ** - возможность переноса программы на другую ЭВМ.

**Модем** - устройство связи компьютеров для передачи файлов по каналам связи.

**Мышка** - приспособление, которое можно перемещать по столу и которое подсоединено к ЭВМ.

**Объем понятия** - совокупность вещей или явлений, отвечающих данному понятию.

**Операционная система** - главная управляющая программа на ЭВМ.

**Отказ** - ситуация, когда прекращается выполнение программы.

**Отладка программ** - процесс поиска и исправления ошибок в про­граммах на ЭВМ.

**Паскаль** - язык структурного программирования.

**Персональные компьютеры** - малогабаритные вычислительные ма­шины.

**Поиск данных** - отбор данных по определенной комбинации при­знаков.

**Почтовый сервер** - компьютер, выделенный для переписки по элект­ронной почте.

**Постановка задачи -** четкое выделение в задаче требуемого и исход­ного.

**Правила в Прологе** - правила логического вывода.

**Правильность программ** - правильность результатов, получаемых при выполнении программ на ЭВМ.

**Предикаты** - математические высказывания о некоторых объектах или их свойствах.

**Представление данных** - подбор массивов, переменных и других структур для размещения данных в памяти ЭВМ.

**Принтеры** - печатающие устройства, подсоединяемые к компьютерам.

**Программа для ЭВМ** - форма представления данных и команд, пред­назначенных для получения определенных результатов или спо­соба функционирования ЭВМ.

**Программа** правильная, если она дает правильные результаты для любых допустимых исходных данных.

**Программа содержит ошибки,** если ее выполнение приводит к отка­зам, сбоям или получению неправильных результатов.

**Пролог** - система искусственного интеллекта, способная воспроиз­водить логические выводы и умозаключения.

**Процессор** - устройство управления компьютеров и автоматических приборов.

**Псевдокод** - язык для описания логики работы программ с исполь­зованием лексики родного (русского) языка.

**Разработка сверху-вниз** - способ последовательной разработки про­ектов от основной задачи к решению вспомогательных подзадач.

**Региональная вычислительная сеть** - сеть, связывающая компьютеры в пределах определенного региона.

**Редакторы текстов** - программы подготовки и редактирования текс­тов на ЭВМ.

**Результат неправильный,** если он противоречит требованиям задачи.

**Результат правильный,** если он отвечает поставленным требованиям.

**Результирующие данные** - результаты обработки данных.

**Реляционные базы** - базы данных, в которых информация хранится в форме таблиц.

**Сбой** - потеря части данных либо получение непредусмотренных данных.

**Сервер** - компьютер, подключенный к сети и используемый для хранения информации.

**Сетевые программы** - программы приема и передачи данных в сетях ЭВМ.

**Электронная библиотека** - совокупность электронных книг, разме­щенных на одном или нескольких серверах.

**Электронные книги** - гипертексты, размещаемые на серверах и до­ступные для чтения.

**Электронная почта -** способ передачи писем с помощью ЭВМ и средств связи.

**Электронные таблицы** - программы для выполнения и хранения калькуляций на ЭВМ.

**Электронные технологии** - информационные технологии, связанные с представлением информации в электронном виде.

**Электронные учебники** - программы с учебными материалами и тестами по определенному предмету.

**Эффективность программ** - минимальность времени выполнения программ на ЭВМ.

**Языки программирования** - языки для записи программ для ЭВМ.

**Языки спецификаций** - языки для записи спецификаций.

**Excel** - электронные таблицы на компьютерах IBM PC с операци­онной системы Windows .

**IBM PC** - наиболее распространенные персональные компьютеры.

**Macintosh** - самые удобные персональные компьютеры.

**MS DOS** - операционная система для компьютеров IBM PC.

**Quick Basic** - язык структурного программирования для компьюте­ров IBM PC с операционной системой MS DOS.

**Visual Basic** - объектно-ориентированный язык программирования для IBM PC с операционной системой Windows.

**Windows** - операционная система для старших моделей компьюте­ров IBM PC.

**Word** - редактор текстов для операционной системы Windows.

**WWW** (world-wide web) - всемирная распределенная сеть электрон­ных библиотек.

# Список рекомендуемой литературы

1. Государственный образовательный стандарт высшего профес­сионального образования. - М.: Госкомвуз России, 1995.

2. *Каймин В.А.* Информатика. Учебник для вузув. - М.: «Высшее образование», 1998.

3. Информатика. Учебник/Под ред.Н.В. Макаровой. - М.: Фи­нансы и статистика, 1997.

4. *Каймин В.А.* Информатика//Программы авторских курсов по гуманитарным и социально-экономическим дисциплинам для высшей школы. - М.: Госкомвуз, 1996.

5. *Каймин В.А.* Информатика и дистанционное обучение. - М.: Научная книга, 1996.

6. *Каймин В.А.* Информатика. Учебник для дистанционного обу­чения. М., МЭСИ, 1999.

7. Дистанционное образование в России: состояние и перспекти­вы. - М.: МЭСИ, 1998.

8. *Евреинов Э.В., Каймин В.А.* Информатика и дистанционное об­разование. - М., 1998.

9. *Лобачев С.Л., Солдаткин В.И.* Дистанционные образовательные технологии: информационный аспект. - М.: МЭСИ, 1998.

10. Информатика. Энциклопедический словарь для начинающих/ Сост. Д.А. Поспелов. - М.: «Просвещение», 1994.

11. *Глушков В.М.* Основы безбумажной информатики. - М.: На­ука, 1982.

12. *Брой М.* Информатика. Основополагающее введение. - М.: ДИАЛОГ-МИФИ, 1998.

13. *Байков В.* ИНТЕРНЕТ: от E-MAIL к WWW в примерах. -СПб.: BHV, 1996.

14. *Артамонов Б.Н.* и др. Основы современных компьютерных технологий. - Спб.: КОРОНА, 1998.

15. *Фигурнов Э.В.* IBM PC для пользователя. - М.: ИНФРА-М, 1999.

16. *Ершов А.П.* и др. Основы информатики и вычислительной техники. Пробный учебник для 10-11-х классов. - М.: Просвеще­ние, 1985.

17. *Каймии В.А.* и др. Основы информатики и вычислительной техники. Пробный учебник для 10-11-ых классов. - М.: Просвеще­ние, 1989.

18. *Гейн А.Г.* и др. Основы информатики и вычислительной тех­ники. Пробный учебник для 10-11 классов. - М.: Просвещение, 1996.

19. *Семакин И.Г.* и др. Информатика. Учебник для 7-9 классов. - М.: БИНОМ, 1998.

20. *Каймин В.А.* и др. Информатика. Пособие для поступающих в вузы. - М.: Бридж, 1994.

21. *Шафрин Ю.А.* Основы компьютерной технологии. Учебное пособие для учащихся старших классов. - М.: ABF, 1996.

22. *Громов Г.Р.* Очерки информационной технологии. - М.: Инфоарт, 1993.

22. *Каймин В.А.* Основы компьютерной технологии. - М.: Фи­нансы и статистика, 1992.

23. *Лихачева Г.Н.* Информационные технологии в экономике. - М.: МЭСИ, 1999.

24. *Кураков Л.П., Смирнов С.Н.* Информация как объект правовой защиты. М., Гелиос, 1998.

25. *Крылов В.В.* Информационные компьютерные преступления. М., ИНФРА-М, 1997.

26. *Терещенко Л.К. и др.* Защита прав создателей и пользователей программ для ЭВМ и баз данных. М.: МЮ РФ, 1996.

ОГЛАВЛЕНИЕ

Введение 2

Глава 1. ИНФОРМАЦИЯ И ПЕРСОНАЛЬНЫЕ ЭВМ 3

1.1. Введение в информатику 3

1.2. Персональные компьютеры 6

1.3. Работа на персональных ЭВМ 9

1.4. Редактирование текстов на ЭВМ 11

1.5. Контроль знаний на ЭВМ 13

Глава 2. ЭЛЕМЕНТЫ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ 17

2.1. Основные возможности Интернет 17

2.2. Базы данных на ЭВМ 20

2.3. Выполнение расчетов на компьютерах 23

2.4. Постановка и решение задач 25

Глава 3. ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА 28

3.1. Элементы математической логики 28

3.2. Основные логические операции 30

3.3. Элементы языка Пролог 33

3.4. Базы знаний на ЭВМ 37

3.5. Законы логического вывода 39

Глава 4. ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ 45

4.1. Основные свойства алгоритмов 45

4.2. Базовые средства программирования 49

4.3. Основы структурного программирования 53

4.4. Основы безошибочного программирования 58

4.5. Средства обработки данных 61

Глава 5. ТЕХНОЛОГИЯ РЕШЕНИЯ ЗАДАЧ 67

5.1. Решение задач на ЭВМ 67

5.2. Анализ правильности алгоритмов 74

5.3. Решение прикладных задач 80

5.4. Элементы доказательного программирования 88

5.5. Решение сложных задач 95

Глава 6. ЭКЗАМЕНЫ ПО ИНФОРМАТИКЕ 103

6.1. Экзамены и зачеты по информатике 103

6.2. Решение экзаменационных задач 106

6.3. Проверка программ на ЭВМ 112

6.4. Олимпиадные задачи по информатике 116

6.5. Технология дистанционного обучения 127

Приложение 130

ИНТЕРПРЕТАТОР ЯЗЫКА ПРОЛОГ 130

1. Назначение интерпретатора Пролога 130

2. Запуск интерпретатора Пролога 130

3. Диалог с программами на Прологе 131

4. Ввод и редактирование программ 131

5. Операции с файлами 132

6. Краткое описание языка Пролог 133

6.1. Базовые понятия языка Пролог 133

6.2. Описания фактов 134

6.3. Правила записи вопросов 135

6.4. Запись правил и процедур 135

6.5. Встроенные предикаты Пролога 136

Толковый словарь 138

Список рекомендуемой литературы 142