Ответы экзамена по предмету "Языки программирование и информатика"

1. Понятие информатики - разные подходы к определению.

Термин «информатика» происходит от французского слова Informatique, полученного комбинацией двух слов: Information (информация) и Automatique (автоматика). Французское словосочетание вкладывает в термин «информатика» смысл автоматической обработки информации.

Информатика — это наука об информации, способах ее сбора, хранения, обработки и представления с помощью электронных средств. Таким образом, информатика имеет дело с информационными процессами, как обобщенно называют получение, хранение, обработку и представление информации, происходящими в объектах произвольной природы: технических, биологических, социальных. Экономическая информатика — это наука об информационных системах, применяющихся для подготовки и принятия решений в управлении, экономике и бизнесе, а также об экономике этих систем.

Весьма авторитетный специалист в области информатики Кристен Нюгард (Kristen Nygaard) дает такое определение: информатика — это наука об информационных процессах и связанных с ними явлениях в обществе, природе и человеческой деятельности1. В соответствии с этим определением поле деятельности информатики гораздо шире, чем то, что обычно изучает наука о компьютерах (computer science). Занимаясь определенным классом явлений, информатика не может претендовать на то, чтобы быть строгой наукой подобно математике. Явления, возникающие в связи с информационными процессами, могут быть самой различной природы и должны изучаться с позиций соответствующей науки, как это делается в физике, экономике, социологии, биологии и т.д . Известно, что далеко не все в этих науках описывается строгими математическими законами и наука не заканчива- ется там, где исчерпывают себя математические методы. К таким наукам относится и информатика.

Экономическая информатика — это наука об информационных системах, применяющихся для подготовки и принятия решений в управлении, экономике и бизнесе, а также об экономике этих систем.

Термин «информационные системы» вбирает в себя два важных понятия — «информация» и «системы». Каждое из них настолько важно, что для их описания существуют целые теории: теория информации и теория систем. Мы лишь познакомимся с базовыми понятиями, необходимыми для дальнейшего изложения.

таким образом информатика изучает:

- информацию и ее свойства.
- информационные процессы:
    - хранение;
    - обработка;

- передача;

с помошью компьютерной технологии.

2. Информатика и ее структура.

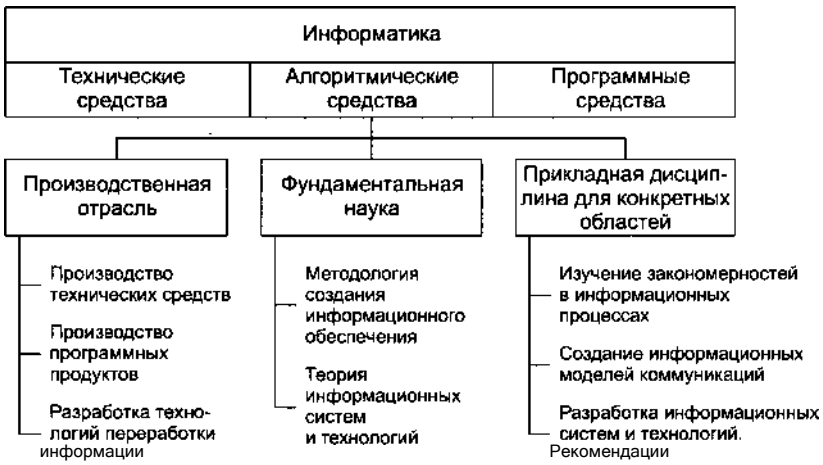
Термин *информатика* возник в 60-х гг. во Франции для названия области, занимающейся автоматизированной обработкой информации с помощью электронных вычислительных машин. Французский термин *informatigue* (информатика) образован путем слияния слов *information* (информация) и *automatigue* (автоматика) и означает буквально информационную автоматику, или автоматизированную переработку информации. В англоязычных странах этому термину соответствует синоним *computer science* (наука о компьютерной технике).

Информатика занимается изучением процессов преобразования и создания новой информации более широко, практически не решая, в отличие от кибернетики, задачи управления различными объектами. Поэтому может сложиться впечатление об информатике как о более емкой дисциплине, чем кибернетика. Однако информатика не занимается решением проблем, не связанных с использованием компьютерной техники, что, несомненно, сужает ее, казалось бы, обобщающий характер. Между этими двумя дисциплинами провести четкую границу не представляетсявозможным в связи с ее размытостью и неопределенностью, хотя существует довольно распространенное мнение о том, что информатика является одним из направлений кибернетики.

Информатика в широком смысле представляет собой единство разнообразных отраслей науки, техники и производства, связанных с переработкой информации главным образом с помощью компьютеров и телекоммуникационных средств связи во всех сферах человеческой деятельности.

Информатику в узком смысле можно представить как состоящую из

трех взаимосвязанных частей — технических (hardware), программных (software) и алгоритмических средств (brainware). В свою очередь,



информатику в целом и каждую ее часть обычно рассматривают с разных позиций (рис. 2.3):

Структура информатики как отрасли, науки, прикладной дисциплины.

как производственную отрасль, как фундаментальную науку, как прикладную дисциплину.

Информатика как производственная отрасль состоит из однородной совокупности предприятий разных форм хозяйствования, где занимаются производством компьютерной техники, программных продуктов и разработкой современной технологии переработки информации. Специфика и значение информатики как отрасли производства состоят в том, что от нее во многом зависит рост производительности труда в других отраслях народного хозяйства. Более того, для нормального развития этих отраслей производительность труда в самой информатике должна возрастать более высокими темпами, так как в современном обществе информация все чаще выступает как предмет конечного потребления: людям необходима информация о событиях, происходящих в мире, о предметах и явлениях, относящихся к их профессиональной деятельности, о развитии науки и самого общества. Дальнейший рост производительности труда и уровня благосостояния возможен лишь на основе использования новых интеллектуальных средств и человеко-машинных интерфейсов, ориентированных на прием и обработку больших объемов мультимедийной информации (текст, графика, видеоизображение, звук, анимация). При отсутствии достаточных темпов увеличения производительности труда в информатике может произойти существенное замедление роста производительности труда во всем народном хозяйстве. В настоящее время около 50 % всех рабочих мест в мире поддерживается средствами обработки информации.

Информатика как *фундаментальная наука* занимается разработкой методологии создания информационного обеспечения процессов управления любыми объектами на базе компьютерных информационных систем. Существует мнение о том, что одна из главных задач этой науки — выяснение, что такое информационные системы, какое место они занимают, какую должны иметь структуру, как функционировать, какие общие закономерности им свойственны. В Европе можно выделить следующие основные научные направления в области информатики: разработка сетевой структуры, компьютерно-интегрированные производства, экономическая и медицинская информатика, информатика социального страхования и окружающей среды, профессиональные информационные системы.

Цель фундаментальных исследований в информатике — получение обобщенных знаний о любых информационных системах, выявление общих закономерностей их построения и функционирования.

Информатика как *прикладная дисциплина* занимается:

- изучением закономерностей в информационных процессах (накопление, переработка, распространение информации);
- созданием информационных моделей коммуникаций в различных областях

человеческой деятельности;

- разработкой информационных систем и технологий в конкретных областях и выработкой рекомендаций относительно их жизненного цикла: этапов проектирования и разработки, производства, функционирования и т. д.

Главная функция информатики заключается в разработке методов и средств преобразования информации и их использовании с целью организации технологического процесса переработки информации.

Задачи информатики состоят в следующем:

- исследование информационных процессов любой природы;
- разработка информационной техники и создание новейшей технологии переработки информации на базе полученных результатов исследования информационных процессов;
- решение научных и инженерных проблем создания, внедрения и обеспечения эффективного использования компьютерной техники и технологии во всех сферах общественной жизни.

3. Новые информационные технологии.

Информационные технологии (ИТ) — инфраструктура, обеспечивающая реализацию информационных процессов. Ее формируют: каналы связи, по которым передается информация; программы, управляющие сбором, хранением, обработкой и представлением информации; компьютеры и аппаратура, обеспечивающие выполнение этих программ. Информация на технологическом уровне выступает в виде данных, поэтому сами данные также относятся к ИТ. К информационным технологиям также относятся инструменты, с помощью которых реализуются информационные процессы: языки, используемые для написания программ, протоколы, применяющиеся при передаче данных, модели хранения и передачи данных и т.д.

Итак, инфраструктура ИТ — это, с одной стороны, аппаратно-программная среда и телекоммуникации, а с другой — инструментарий осуществления информационных процессов. В центре инфраструктуры — данные.

ИТ иногда рассматривают как «компьютерную инфраструктуру» организации. Составляющие ИТ имеют «твердую основу» {hardware) в виде компьютеров, сетей, телекоммуникационного оборудования и каналов связи, а также «мягкую основу» {software) в виде программного обеспечения, управляющего работой аппаратуры. Для нас важнейшей составляющей ИТ, своеобразным ядром являются данные, организованные с помощью специальных программ в базы данных.

База данных (БД) — это организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность данных, используемая для удовлетворения информационных потребностей пользователей. БД — это своеобразная среда, в которой данные превращаются в информацию. Базы данных, по существу, представляют собой важнейшее связующее звено, мост, соединяющий техническую мощь

информационных систем с реальными задачами конкретных ФП и бизнес-приложений.

Информационные технологии обеспечивают информационные процессы вне зависимости от их содержания, они одинаково работают с бухгалтерскими сведениями и сводками погоды — обеспечивают хранение, получение информации и передачу ее средствами телекоммуникаций от отправителя по конкретному адресу в заданном виде. Результат работы ИТ — обработка и доставка сообщения, содержание которого ИТ не интересует, от отправителя — к получателю.

Передовая практика управления утверждает, что основная роль ИТ в организации — информационное обслуживание ее подразделений. Поскольку термин «информационное обслуживание» весь-ми широк, его следует уточнить. Под *информационным сервисом* мы **будем** понимать информационный процесс, функционирующий в режиме, определяемом одним или несколькими бизнес-процессами или проектами. Информационный сервис, выполняемый средст-нпми ИТ, мы будем далее именовать *сервисом ИТ*. Попросту говоря, информационный сервис — это информационный процесс, параметры которого значимы для протекания бизнес-процессов или проектов в организации. При этом не каждый информационный1Трнис выполняется средствами ИТ. Например, если юрист отыски-ii.li-т и твердой копии некоторого кодекса необходимую для данного Влучая статью закона, речь идет об информационном сервисе, который не является сервисом ИТ, — информационные технологии в этом случае не используются. Напротив, расчет баланса в 1С или любой другой бухгалтерской системе является сервисом ИТ.

4. „ Новая экономика " - экономика информации.

Информационные системы изменили облик мировой экономики. Появились новые термины, определяющие экономику, работа которой характеризуется глубоким проникновением ин формационных систем: интернет-экономика, новая экономика, метакапитализм.

Так что будем говорить об экономике, базирующейся на современных достижениях информационных технологий и информационных систем, как об «экономике современного информационного общества». Рассмотрим несколько характерных черт гакой экономики'.

1. Продукция некоторых отраслей экономики в динамике характеризуется повышением качества и производительности при относительном снижении цены.

2. Происходит декапитализация компаний: крупные компании дробятся и превращаются в конгломерат тесно связанных, но до статочно независимых небольших фирм-партнеров, работающих пи главную компанию — носитель торговой марки или брэнда

3. Развитие прямых связей с потребителями, работа по их индивидуальным заказам.

В заключение очертим ближайшую перспективу развития эко номики и общества, к которой приведет изменяющаяся роль информации. Используем и разовьем для этого некоторые тезм сы одного из первых исследователей постиндустриального, или информационного, общества Д. Белла', которые были сформулп рованы им уже давно, но не утратили актуальности, а также П. Дракера[2]:

1) центральное место в информационном обществе принад лежит знаниям;
2) на основе знаний быстрыми темпами создаются и приме няются в деятельности компаний новые интеллектуальные тех нологии;
3) отвечая на рост интеллектуальных технологий, быстро ра стет класс людей, обладающих необходимыми знаниями;
4) власть постепенно переходит к меритократии, т.е. высоко образованным профессионалам;
5) основным ресурсным ограничением является нехватка ин формации;
6) экономика информационного общества характеризуется как экономика информации;
7) изменяется характер труда: все чаще становится трудно определить — работает ли сотрудник в данный момент. Исчезает необходимость примитивного контроля занятости — остается контроль по результатам. Вместе с тем отпадает нужда в постоян ном присутствии сотрудника в офисе в рабочее время. Иногда отпадает необходимость и в физическом офисе — достаточно ло гического или виртуального;
8) существенно меняется роль женщин и мужчин в организа ции семейной жизни: возможность организовать работу вне зара нее определенного «присутственного места» разрушает привыч ную схему распределения ролей в семье. Виртуальные организа ции с удаленным доступом к трудовым ресурсам позволяют женщине уделять больше внимания работе, а мужчине — семье. Гораздо большее количество специалистов получают возможность заниматься творческим трудом и непосредственно участвовать в создании стоимости продуктов.
9) Очевидна главная мысль — знания становятся основным ресурсом прогрессивного развития не только индивида, но и бизнеса, экономики компаний, отрасли. Для будущих молодых специалистов это означает, что обладающий глубокими знаниями выпускник университета имеет больший стартовый капитал и серьезное конкурентное преимущество на современном рынке труда.

5. Данные - информация - знания.

Данные — это фиксированные сведения о событиях и явлениях. Будучи полученными, данные могут храниться произвольное время в неизменном виде. Информация — это обработанные данные, представленные в виде, пригодном для принятия решений получателем. Информация появляется при необходимости решить конкретную задачу или ответить на запрос на основании имеющихся данных. Примерами данных могут служить: линии, оставляемые на бумажной ленте самописцами (например, кардиограмма или кривая данных сейсмографа); рентгеновский снимок части тела больного; колонки чисел, представляющие котировки акций на бирже.

Информация — это обработанные данные, которые представлены в виде, пригодном для принятия получателем решений или проведения аналитических исследований.

Знания — это обработанная информация, использованная и используемая для принятия решений и решения задач, а также сведения о способах обработки информации для приведения ее к виду, пригодному для принятия решений.

В информационных системах можно найти примеры информации всех видов. Данные, как информация с первичных документов (счетов, накладных и т.д.), заносятся в базу данных и хранятся тоже как данные. Сформированный запрос инициирует поиск необходимых данных в базе, их обработку и представление в заданном виде получателю уже в качестве информации.

6. Свойство данных, информации и знаний

Свойства данных. Репрезентативность данных означает способность собранных данных адекватно отобразить свойства описываемого ими явления. Важное значение здесь имеют:

- правильный отбор объектов для сбора данных;
- определение набора существенных признаков для измерения;
- достаточное количество объектов;
- соответствие данных формулировке задачи, для решения ко торой осуществляется сбор данных.

Точность данных. Принято выделять:

- формальную точность, измеряемую значением единицы по следнего разряда числа;
- реальную точность, измеряемую значением единицы послед него разряда, достоверность которого гарантируется;
- максимальную, или достижимую, точность, которая может быть получена при конкретных условиях сбора данных;
- необходимую точность, определяемую требованиями задачи, для решения которой данные собираются.

Достоверность данных — это способность представлять опи сываемые объекты с заданной по условиям решаемой задачи точ ностью.

Свойства информации.

Актуальность информации определяется степенью сохранения ее пригодности для принятого решения и зависит от того, в течение какого периода времени сохраняется репрезентативность данных, использованных для получения информации. Если данные устаревают или теряют способность представлять описываемый ими процесс, говорят о необходимости «актуализации» данных, т.е. их обновления.

Своевременность информации означает, что она получена в нужный момент принятия решения, без опоздания. Информация, поступившая после принятия решения, скорее всего, не нужна. С другой стороны, не всякая заблаговременность предоставления информации хороша: может быть утрачена ее актуальность.

Свойства знаний. Как мы знаем, знания формируются из обработанной информации, используемой и уже использовавшейся для принятия решений, так что положительные свойства и данных, и информации в знаниях молчаливо предполагаются. Поэтому знания характеризуются свойствами несколько иного типа.

Знания могут существовать в следующих видах:

- предметный или конкретный, использующий информацию из конкретной области, — это методики принятия решений для конкретно поставленной задачи (например, как спилить де рево, как обработать древесину, как сделать лодку и весла, как плыть на лодке по реке и т.д.);
- концептуальный, или обобщающий, использующий инфор мацию из многих областей и определяющий, как извлекать знания из информации, — это методологии (например, прин ципы кораблестроения, землеведения, управления компания ми и т.д.);
- метазнания (знания о знаниях) — генерируют новые знания.

Примеры метазнаний в конкретной области:

химия — таблица Менделеева, предсказавшая появление новых химических элементов; генетика — теория гомологических рядов Н.И. Вавилова, предсказывающая наличие растений с данными свойствами в конкретном районе.

7. Понятие энтропии системы

Для определения меры информации необходимо ввести понятие меры неопределенности. Неопределенность — непременное свойство любого хозяйственного или управленческого решения: такие решения — это выбор из нескольких возможных вариантов, и полной уверенности, что выбран действительно лучший, практически никогда не бывает. Даже в простой ситуации, выходя утром из дома, мы принимаем решения о том, как лучше одеться и взять ли зонт: существует опасность промокнуть, если будет обещанный по прогнозу дождь. Уменьшение неопределенности выбора лучшего решения возможно благодаря получению новых сведений или дополнительной информации.

Принятой мерой неопределенности системы ос является энтропия, обозначаемая Н(α). При получении сообщения β энтропия системы — Нβ(α). Как мы заметили, может быть, что Нβ(α). < Н(α), Нβ(α) > Н(α) и Нβ(α) = Н(α) , — все зависит от того, что за сообщение р получено. Интересно, что именно разность Нβ(α) - Н(α) оказывается важной характеристикой полученного сообщения р.

Этой важной характеристикой сообщения р о системе а является количество информации Iβ(α) , содержащееся в сообщении β о системе ос:

Iβ(α)=Н(α) - Нβ(α)

Понятно, что величина Iβ(α) может быть положительной (когда сообщение уменьшает неопределенность), отрицательной (когда неопределенность растет) и нулевой (когда сообщение не несет информации, полезной для принятия решения). В последнем случае Нβ(α) = Н(α), т.е. неопределенность системы по получении сообщения р не изменилась, и количество информации в Р равно нулю. Другим крайним случаем является ситуация, когда сообщение βРполностью снимает неопределенность и Нβ(α) = 0. Тогда сообщение β содержит полную информацию о системе α и Iβ(α)=Н(α). Теперь для определения количества информации нам надо понять, как оценивать энтропию системы. В общем случае энтропия системы, имеющей η возможных состояний (Н(α)), согласно формуле Шеннона равна

H(α)= -ΣPílogPí

где Pí — вероятность того, что система находится в i-u состоянии.

8. Количественная мера информации. Формула Шеннона

Вывод формулы Шеннона

Нам необходимо научиться оценивать степень неопределенности различных ситуаций, опытов. Для самых простых опытов, имеющих к равновероятных исходов, степень неопределенности измеряется с помощью самого числа к: при к = 1 никакой неопределенности нет, так как исход предопределен, но не случаен. При росте числа возможных исходов предсказание результата опыта становится все более затруднительным, так что естественно предположить, что мера степени неопределенности является функцией k — j[k], причем/I) = 0 nj[k] монотонно растет с ростом к.

Кроме того, надо научиться оценивать неопределенность нескольких опытов. Рассмотрим два независимых опыта аир (т.е. таких два опыта, что любые сведения об исходе первого никак не меняют вероятностей исходов второго). Если опыт а имеет р равновероятных исходов, а опыт р — q равновероятных исходов, то сложный опыт ар, состоящий в одновременном выполнении опытов аир, очевидно, обладает большей неопределенностью, чем каждый опыт а или р в отдельности.

Клод Шеннон в 1950 г. предложил в качестве меры неопределенности системы а с к состояниями энтропию Н(а):

H(α)= -ΣPílogPí

где Pí — вероятность того, что система находится в i-u состоянии.

Энтропия равна нулю только в одном случае, когда все вероятности P, равны нулю, кроме одной, которая равна единице. Это точно описывает отсутствие неопределенности: система находится всегда в одном и том же состоянии.

Например, энтропия нашего алфавита из 32 букв: H— log 32 = = 5 бит. Энтропия десятичного набора цифр: H= log 10 = 3,32 бита. Энтропия системы, в которой отдельно хранятся 32 буквы и 10 цифр: H= log (32 • 10)= 5 + 3,32 = 8,32 бита.

9. Синтаксическая мера информации

Синтаксическая мера информации отображает структурные характеристики информации и не затрагивает его смыслового содержания. На синтаксическом уровне учитываются тип носителя, способ представления информации, скорость предачи и т.д.

В качестве синтаксической меры количество информации представляет объем данных.

Объем данных V_d в сообщении (измеряется количеством символов (разрядов) в этом сообщении. Как мы упоминали, в двоичной системе счисления единица измерения — бит. На практике наряду с этой «самой мелкой» единицей измерения данных чаще применяется более крупная единица — байт, равная 8 бит. Для удобства в качестве измерителей используются кило- (10^3), мега-(10^6), гига- (10^9) и тера- (10^{12}) байты и т.д. В знакомых всем байтах измеряется объем кратких письменных сообщений, толстых книг, музыкальных произведений, изображений, а также программных продуктов. Понятно, что эта мера никак не может характеризовать того, что и зачем несут эти единицы информации.

Синтаксическая мера информации определяет отношения информации и технологии, семантическая — информации и получателя.

10.Семантическая мера информации.

Итак, одной синтаксической меры информации явно недостаточно для характеристики сообщения: в нашем примере с погодой в последнем случае сообщение приятеля содержало ненулевой объем данных, но в нем не было нужной нам информации. Заключение о полезности информации следует из рассмотрения содержания сообщения. Для измерения смыслового содержания информации, т.е. ее количества на семантическом уровне, введем понятие «тезаурус получателя информации».

Тезаурус — это совокупность сведений и связей между ними, которыми располагает получатель информации. Можно сказать, что тезаурус — это накопленные знания получателя.

В очень простом случае, когда получателем является техническое устройство — персональный компьютер, тезаурус формируется «вооружением» компьютера — заложенными в него программами и устройствами, позволяющими принимать, обрабатывать и представлять текстовые сообщения на разных языках, использующих разные алфавиты, шрифты, а также аудио- и видеоинформацию из локальной или всемирной сети. Если компьютер не снабжен сетевой картой, нельзя ожидать получения на него сообщений от других пользователей сети ни в каком виде. Отсутствие драйверов с русскими шрифтами не позволит работать с сообщениями на русском языке и т.д. Если получателем является человек, его тезаурус — это тоже своеобразное интеллектуальное вооружение человека, арсенал его знаний.

Относительной мерой количества семантической информации служит коэффициент содержательности С, который определяется как отношение количества семантической информации к ее объему данных Vd, содержащихся в сообщении β

$$C=Ic/V_д$$

11.Системы счисления.

Числа отображаются с помощью системы счисления.

Совокупность символов, при помощи которых записывается система счисления, называется *алфавитом* системы счисления. Количество символов, составляющих алфавит, называется его *размерностью.*

Системы счисления делятся на позиционные и непозиционные. В *позиционной системе счисления* значение каждой цифры в записи числа зависит от ее позиции (разряда). В *непозиционной системе счисления* для обозначения чисел вводятся специальные знаки, количественное значение которых всегда одинаково и не зависит от их места в записи числа, например, числу 25 в непозиционной римской системе счисления соответствует запись XXV, а числу 53 соответствует запись ЫН.

Последовательность чисел, каждое из которых задает «вес» соответствующего разряда, называется *базисом* позиционной системы счисления. Основным достоинством позиционной системы счисления является возможность записи произвольного числа при помощи ограниченного набора символов.

В табл. 11.4 сравниваются несколько двоичных и десятичных чисел, находящихся в одной и той же позиции.Обе эти системы счисления являются позиционными. Значение каждой цифры в числе легко определить, воспользовавшись формулой

У-В".

Здесь *V*— значение, *В* — основание системы счисления, ЛГ— порядковый номер позиции.

Двоичные и десятичные числа

Двоичные			Десятичные	
Двоичное (двоичное представлен	**Двоичное (десятичное представлен**	**Степень**	**Десятичное число (десятичное**	**Степ**
1	**1**	2°	**1**	**10°**
10	2	*2′*	**10**	**10^{1}**
100	4	2^{2}	**100**	ю^{2}
1000	8	2^{3}	**1000**	**10^{3}**
10000	16	2^{4}	**10000**	**10^{4}**
100000	32	2^{5}	**100000**	ю^{5}
1000000	64	2^{6}	**1000000**	**10^{6}**

Очевидно, что десятичная запись намного удобнее двоичной, поскольку более компактна. Так, в 6-й позиции в десятичной системе закодирован миллион, а в двоичной — только шестьдесят четыре. В двоичной системе представление чисел получается весьма громоздким, но оно идеально для компьютера, так как не требует сложных инженерных и технологических решений.

Позиционную систему счисления называют *традиционной*, если ее базис образуют члены геометрической прогрессии, а значения цифр есть целые неотрицательные числа. Знаменатель *P* геометрической прогрессии, члены которой образуют базис традиционной системы счисления, называется *основанием* этой системы счисления. Традиционные системы счисления с основанием *P* иначе называют *P-ичными.*

Для того чтобы показать, в какой системе счисления записано число, после числа указывают основание в качестве нижнего индекса. Например 10_2 — два в двоичной системе счисления, или 2_{10} — два в десятичной системе счисления.

Если в позиционной системе счисления базис не является геометрической прогрессией или значения цифр могут принимать отрицательные значения, такую систему счисления называют *нетрадиционной* **.** Системы счисления

Любое натуральное число можно записать в Р-ичной системе счисления

единственным образом. В более строгом виде это утверждение можно сформулировать в следующем виде.

Пусть *P* — произвольное натуральное число, большее единицы. Существует единственное представление любого натурального числаX в виде степенного ряда:

$X = a_n P^n + a_{n\_x} P^{\text{п}\sim 1} + \ldots + a_{\ell} P + a_0$, где 0<;я$_{\text{г}}$ <*P*, 0

В Р-ичной системе счисления любое неотрицательное вещественное число, содержащее целую и дробную части, можно записать в виде

| Десятеричная |

| Шестнадцатеричная]

Факториальная

Фибоначчиева

$a = a_nP^n + a_{n\_}\backslash P^n{\sim}^x + ... + a{\pm}P + a_0 + a_{\_x}P{\sim}^{\{} + a_{\_2}P{\sim}^2 + ... =$
$= {\wedge}й_гP^г, 0 <> a_{\{}<P, П'га\ 0.$
Здесь P > 0 — основание позиционной системы счисления, — цифры числа *а* в Р-ичной системе счисления.

Для записи чисел в Р-ичных системах счисления используют десятичные цифры (О... 9), которые дополняются буквами латинского алфавита.

В табл. 11.5 приведен пример того, как будет выглядеть число 25, записанное в разных Р-ичных системах счисления.
Число 25, записанное в разных Р-ичных системах счисления

Система счисления	**Основание**	**Размерность алфавита**	**Алфавит**	**Пример**
Десятичная	10	10	0,1, 2,3,4, 5, 6, 7,8,9	$25 = 2 \times 10^1 + 5 \times 10^\circ$
Двоичная	2	2	0,1	$11001 = 1x2^4+1x2^3 + + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^\circ$
Троичная	3	3	0, 1,2	$221 = 2 \times 3^2 + 2 \times 3^1 + 1 \times 3^\circ$
Восьмеричная	8	8	0, 1,2,3,4,5, 6,7	$31 = 3 \times 8^1 + 1 \times 8^\circ$
Шестнадцатеричная	16	16	0,1, 2,3,4, 5, 6, 7, 8, 9, A, B, C, Э, E, E	$19 = 1 \times 16^1 + 9x\ 16^\circ$

Для записи отрицательных чисел в Р-ичных системах счисления используют знак «минус», помещенный перед числом. При записи вещественных чисел для отделения дробной части числа от целой ставится запятая. Таким образом на запись Р-ичных чисел распространяются привычные нам по десятичной системе счисления правила, например: — $2{,}2_{10}$, -P,$0E_{16}$, $-101{,}001_2$.

12. Кодирование информации.

Вся информация, которую хранит, обрабатывает и передает по сетям компьютер, представлена в виде двоичных чисел. Существуют международные стандарты и методы кодирования текстовой, числовой, изобразительной, звуковой и видеоинформации. Знание основных кодовых таблиц очень важно для правильного чтения информации Интернета, электронной почты, текстовых документов в кодировке различных операционных систем.

Для кодирования букв и других символов, используемых в печатных документах, необходимо закрепить за каждым символом числовой номер - код.

В англоязычных странах используются 26 прописных и 26 строчных букв (A ... Z, a ... z), 9 знаков препинания (., : ! " ; ? ( )), пробел, 10 цифр, 5 знаков арифметических действий (+,-,*, /, ^{Л}) и специальные символы (№, %, _, #, $, &, >, <, |, \) - всего чуть больше 100 символов. Таким образом, для кодирования этих

символов можно ограничиться 7-разрядным двоичным числом (от 0 до 1111111, в десятичной системе счисления - от 0 до 127).

Первой такой 7-разрядной кодовой таблицей была ***ASCII*** (American Standard Code for Information Interchange), опубликованная как стандарт в 1963 г. американской организацией по стандартизации American Standards Association (ASA), которая позднее стала именоваться ***ANSI*** (American National Standards Institute, http://www.ansi.org/, поэтому данную кодовую таблицу называют также и ***ANSI).*** Таблица содержала 32 кода команд или управляющих символов (от 0 до 31), большая часть которых сегодня не используется, и 95 кодов (от 33 до 127) для различных знаков, достаточных для работы с английскими текстами, как показано на рисунке 1.1. На рисунке 1.1 символы построчно имеют следующие коды в шестнадцатеричной системе счисления (в скобках - в десятичной):

- 1-я строка с 00 по 0Б и далее с 10 по 1Б(0 - 15, 16 - 31),
- 2-я строка с 20 по 2Б и 30- 3Б (32 - 47, 48 - 63),
- 3-я строка с 40 по 4Б и 50- 5Б (64 - 79, 80 -95),
- 4-я строка с 60 по 6Б и 70- 7Б (96 - 111, 112 -127).

В данной таблице для преобразования прописных букв в строчные достаточно к коду букву прибавить 32 и, наоборот, для преобразования строчных в прописные из кода буквы вычесть 32.

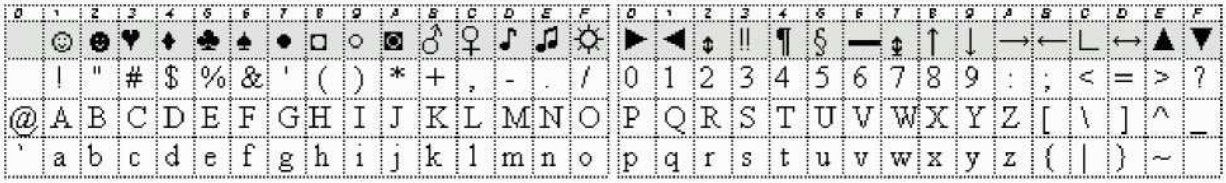


Рисунок 1.1. 7-битная кодовая таблица ***ASCII (ANSI)***

В последующем данная таблица ***ASCII*** была принята как стандарт ведущими международными организациями по стандартизации:

ISO/IEC 646:1991 (ISO - http://www.iso.org/ - International Organization for Standardization и IEC - http://www.iec.ch/ - International Electrotechnical Commission - ведущие международные организации по стандартизации, в области электротехники - совместные стандарты), ***ITU-T Recommendation T.50 (09/92)*** (The International Telecommunication Union - http://www.itu.int/), ECMA-***6*** (European Computer Manufacturers Association).

Однако для нашей страны и многих других стран необходимо было добавить в кодовую таблицу символы национальных алфавитов. Для этого было предложено использовать 8-битную кодовую таблицу, которая могла содержать дополнительно еще 128 символов (с 128 по 255).

В дальнейшем был принят стандарт на 8-битную таблицу

ASCII - ISO/IEC 8859, в которой первые 128 символов оставались те же, что и в 7битной таблице, а символы с 128 по 255 отводились для неанглийских символов.

Существует несколько частей этого стандарта:

- ISO/IEC8859-1:1998**-** Part 1: Latin alphabet No. 1,
- ISO/IEC8859-5:1999**-** Part 5: Latin/Cyrillic alphabet,
- ISO/IEC8859-6:1999**-** Part 6: Latin/Arabic alphabet,
- ISO/IEC8859-7:2003 **-** Part 7: Latin/Greek alphabet,
- ISO/IEC8859-8:1999**-** Part 8: Latin/Hebrew alphabet и т. д.

На рисунке 1.2 представлена вторая половина кодовой таблицы (коды 128-255) для стандарта ***ШО 8859-5.***

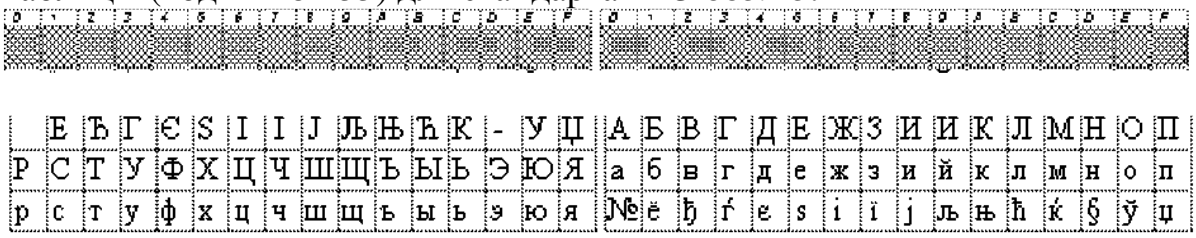


Рисунок 1.2. Кодовая таблица ***ISO 8859-5*** (коды с 128_{10} по 255**ю**)

В 1991 году в Калифорнии была создана некоммерческая организация **Unicode Consortium**, в которую входят представители многих компьютерных фирм (Borland, IBM, Lotus, Microsoft, Novell, Sun, WordPerfect и др.), и которая занимается развитием и внедрением стандарта **«The Unicode Standard»**.

Стандарт кодирования символов **Unicode** становится доминирующим в интернациональных программных многоязычных средах. Microsoft Windows NT и его потомки Windows 2000, 2003, XP, Vista используют **Unicode**, точнее **UTF-16**, как внутреннее представление текста. UNIX- подобные операционные системы типа Linux, BSD и Mac OS X приняли **Unicode** (**UTF-8**), как основное представления многоязычного текста.

Unicode резервируют 1114112 ($2^{20}+2^{16}$) символов кода, в настоящее время используются более 96000 символов. Первые 256 кодов символов точно соответствуют таковым ***ISO 8859-1,*** наиболее популярной 8разрядной таблицы символов «западного мира»; в результате, первые 128
символов также идентичны таблице ***А8СП.***

Числовая информация, как и любая другая, хранится и обрабатывается в компьютерах в двоичной системе счисления - числа представляются в виде последовательностей нулей и единиц.

Существуют два вида чисел и два способа их представления: форма с фиксированной точкой и форма с плавающей точкой. Форма с фиксированной точкой применяется для целых чисел, форма с плавающей точкой - для вещественных (действительных) чисел.

Как это ни странно, не все студенты 1 курса вуза могут ответить на

вопрос, что такое действительные или вещественные числа. Это рациональные и иррациональные числа, у которых может быть как целая, так и дробная часть, записываемая справа от разделителя целой и дробной части. Как разделитель при работе на компьютере раньше всегда использовалась точка, но в современных системах разделитель - точка или запятая - может настраиваться в соответствии со стандартами страны пользователя или с его привычками.

ЭВМ оперирует с числами, содержащими конечное число двоичных цифр (разрядов). Количество разрядов ограничено длиной разрядной сетки машины. Под разрядной сеткой понимается совокупность двоичных разря

дов, предназначенных для хранения и обработки машинных слов (двоичных кодов).

Количество двоичных разрядов и положение запятой в разрядной сетке машины определяют такие важные характеристики ЭВМ, как точность и диапазон представляемых чисел.

Кроме бита и байта, для указания длины формата чисел используется машинное слово, полуслово и двойное слово. Двойное слово и полуслово поразному определяются для разных систем ЭВМ. Кроме того, может использоваться понятие ***тетрада*** - 4 двоичных разряда, которыми может кодироваться, например, одна двоичная цифра.

Двоичные разряды в форматах формируются слева направо (начиная с нулевого разряда).

Кодирование целых чисел

Целые числа в компьютере хранятся в памяти в формате с фиксированной запятой. В этом случае каждому разряду разрядной сетки соответствует всегда один и тот же разряд числа.

Целые числа без знака (положительные) - для их хранения может отводиться последовательность из 8, 16 или 32-х бит памяти. Например, максимальное 8-битное число $A_2 = 11111111_2$ будет храниться следующим образом (прямой код):

1	1	1	1	1	1	1	1

Максимальное значение целого неотрицательного числа достигается в случае, когда во всех ячейках хранятся единицы и равно 2^N-1, где N - разрядность числа.

Для 8-разрядных целых положительных чисел оно будет равно 2^8 - 1 = 255, для 16-разрядных 2^{16} - 1 = 65 535, для 32-разрядных 2^{32} - 1 = 4 294 967 295.

Целые числа со знаком (могут быть положительные и отрицательные) - при их хранении используется последовательность из 8, 16 или 32-х бит памяти, причем старший бит (первый слева) обозначает знак числа - 0 - положительное, 1 - отрицательное. При записи чисел используется не прямой, а дополнительный код двоичного числа равный 2^N - A, где N -

разрядность числа, А - прямой код двоичного числа.

Дополнительным называется код, в котором для положительного числа в знаковом разряде пишется "0", в цифровых - модуль числа, а для отрицательного в знаковом разряде пишется "1", в цифровых - дополнение числа до единицы (инвертирование цифр).

Например, число -1 в 8-разрядном двоичном коде выглядит, как 11111111, -2 - как 11111110 и т. д.

Дополнительный код позволяет заменить арифметическую операцию вычитания операцией сложения, что исключает операцию вычитания их набора команд двоичной арифметики процессора.

Таким образом, при использовании 8-ми разрядов для хранения целых чисел со знаком диапазон их изменения составит от - 128 до 127, если использовать 16 разрядов - от -32 768 до 32 767, 32 разряда - от -2 147 483 648 до 2 147 483 647, что следует учитывать при работе с целыми типами данных при программировании и работе с базами данных.

Кодирование вещественных чисел

Для того чтобы представить действительное число **X** в виде набора целых чисел (двоичных - для представления в компьютерной памяти), его необходимо привести к нормализованной форме:

P

X = ± M • X;

где **M** - ***мантисса*** (дробная часть), **N - основание** системы счисления, а **P** - порядок числа.

Для десятичной системы счисления нормальная форма **X = ± M • 10^P,** для двоичной **X = ± M • 2^P.**

Например, число 22.22_{10} в таком виде будет выглядеть, как

л

+0,2222-10 (при записи чисел в памяти ЭВМ ноль и запятая отсутствуют**).**

Таким образом, действительные число на компьютерах хранится в двоичной системе счисления в виде:

S P M

где **S** - признак знака числа.

13. Принципы Джон фон Неймана, ЭВМ и его структура.

С 1943 г. группа ученых в США начала конструировать вычислительную машину на основе электронных ламп. В 1945 г. к работе был привлечен знаменитый математик Джон фон Нейман. В результате был подготовлен доклад о принципах построения этой машины. Он был опубликован фон Нейманом, и поэтому общие принципы функционирования компьютеров получили название принципов фон Неймана. Первый компьютер, в котором

были воплощены принципы фон Неймана, был построен в 1949 г. английским ученым Морисом Уилксом.

Принципы фон Неймана представляют собой ряд положений, выполнение которых необходимо для эффективной работы вычислительной машины:

- компьютер компонуется из нескольких основных устройств;
- для хранения информации используется специальное запоми нающее устройство;
- данные представлены в запоминающем устройстве в форме двоичных чисел;
- арифметические и логические операции выполняются ариф метико-логическим устройством;
- выполнение программ в вычислительной машине контроли руется устройством управления;
- программа, задающая работу компьютера, хранится в том же запоминающем устройстве, в котором хранятся данные (прин цип хранимой программы);
- для ввода и вывода информации используются отдельные уст ройства ввода-вывода.

Принцип хранимой программы позволяет обрабатывать команды программы так, как если бы они были данными, организуя наиболее эффективное выполнение программ. Большинство современных компьютеров в основных чертах соответствуют принципам, предложенным фон Нейманом.

Структура компьютера – это совокупность его функциональных элементов и связей между ними. Элементами могут быть самые различные устройства – от основных логических узлов компьютера до простейших схем. Кратко сформулируем классические принципы устройства ЭВМ.

Использование двоичной системы счисления для представления чисел. В докладе Неймана были продемонстрированы преимущества двоичной системы для технической реализации узлов компьютера, удобство и простота выполнения в ней арифметических и логических операций. В дальнейшем ЭВМ стали обрабатывать текстовую, графическую, звуковую и другие виды информации, но по-прежнему двоичное кодирование данных составляет информационную основу любого современного компьютера.

Принцип программного управления. Программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности.

Принцип однородности памяти. Программа также должна храниться в виде набора нулей и единиц, причем в той же самой памяти, что и обрабатываемые ей числа. С точки зрения хранения и способов обработки принципиальная разница между программой и данными отсутствует.

Принцип адресности. Структурно основная память состоит из перенумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Адресом ячейки фактически является её номер;

таким образом, местонахождение информации в ОЗУ также кодируется в виде чисел.

14. Персональные компьютеры и их классификация.

Эта категория компьютеров получила особо бурное развитие в течение последних двадцати лет. Из названия видно, что такой компьютер предназначен для обслуживания одного рабочего места. Как правило, с персональным компьютером работает один человек. Несмотря на свои небольшие размеры и относительно невысокую стоимость, современные персональные компьютеры обладают немалой производительностью. Многие современные персональные модели превосходят большие ЭВМ 70-х годов, мини-ЭВМ 80-х годов и микро-ЭВМ первой половины 90-х годов. Персональный компьютер (*Personal Computer, PC)* вполне способен удовлетворить большинство потребностей малых предприятий и отдельных лиц.

Особенно широкую популярность персональные компьютеры получили после 1995 года в связи с бурным развитием Интернета. Персонального компьютера вполне достаточно для использования всемирной сети в качестве источника научной, справочной, учебной, культурной и развлекательной информации. Персональные компьютеры являются также удобным средством автоматизации учебного процесса по любым дисциплинам, средством организации дистанционного (заочного) обучения и средством организации досуга. Они вносят большой вклад не только в производственные, но и в социальные отношения. Их нередко используют для организации надомной трудовой деятельности, что особенно важно в условиях ограниченной трудозанятости.

До последнего времени модели персональных компьютеров условно рассматривали в двух категориях: *бытовые ПК* и *профессиональные ПК*. Бытовые модели, как правило, имели меньшую производительность, но в них были приняты особые меры для работы с цветной графикой и звуком, чего не требовалось для профессиональных моделей. В связи с достигнутым в последние годы резким удешевлением средств вычислительной техники границы между профессиональными и бытовыми моделями в значительной степени стерлись, и сегодня в качестве бытовых нередко используют высокопроизводительные профессиональные модели, а профессиональные модели, в свою очередь, комплектуют устройствами для воспроизведения мультимедийной информации, что ранее было характерно для бытовых устройств.

ДД Под термином *мультимедиа* подразумевается сочетание нескольких видов данных в одном документе (текстовые, графические, музыкальные и видеоданные) или совокупность устройств для воспроизведения этого комплекса данных.

С 1999 по 2002 год в области персональных компьютеров действовали

международные сертификационные стандарты — *спецификации PC99-PC2002*. Они регламентировали принципы классификации персональных компьютеров и оговаривали минимальные и рекомендуемые требования к каждой из категорий. Стандарты устанавливали следующие категории персональных компьютеров:

- *Consumer PC* (массовый ПК);
- *Office PC* (деловой ПК);
- *Mobile PC* (портативный ПК);
- *Workstation PC* (рабочая станция);
- *Entertainmemt PC* (развлекательный ПК).

Каждая категория имела свои особенности: для *портативных ПК* обязательным было наличие средств компьютерной связи, в категории *рабочих станций* предъявлялись повышенные требования к устройствам хранения данных, а в категории *развлекательных**ПК~* к средствам воспроизведения графики и звука.

Одна из целей такой стандартизации состояла и в том, чтобы наметить пути дальнейшего развития и совершенствования персональных компьютеров. Однако развитие аппаратных средств персонального компьютера привело к постепенному размытию границ между разными категориями, а планы развития часто не оправдывались. Поэтому обновление этих стандартов было прекращено, хотя при приобретении компьютера для конкретных задач классификацию, введенную этими стандартами, все еще полезно держать в голове.

Классификация по уровню специализации. По уровню специализации компьютеры делят на *универсальные* и *специализированные.* На базе универсальных компьютеров можно собирать вычислительные системы произвольного состава (состав компьютерной системы называется *конфигурацией).* Так, например, один и тот же персональный компьютер можно использовать для работы с текстами, музыкой, графикой, фото- и видеоматериалами.

Специализированные компьютеры предназначены для решения конкретного круга задач. К таким компьютерам относятся, например, бортовые компьютеры автомобилей, судов, самолетов, космических аппаратов. Бортовые компьютеры управляют средствами ориентации и навигации, осуществляют контроль состояния бортовых систем, выполняют некоторые функции автоматического управления и связи, а также большинство функций по оптимизации параметров работы систем объекта (например, оптимизацию расхода топлива в зависимости от конкретных условий движения объекта). Специализированные мини-ЭВМ, ориентированные на работу с графикой, называют *графическими станциями.* Их используют при подготовке кино- и видеофильмов, а также рекламной продукции. Специализированные компьютеры, объединяющие компьютеры предприятия в одну сеть, называют *файловыми серверами.* Компьютеры, обеспечивающие передачу информации между различными участниками всемирной компьютерной сети, называют *сетевыми серверами.*

Во многих случаях с задачами специализированных компьютерных систем могут справляться и обычные универсальные компьютеры, но считается, что использование специализированных систем все-таки эффективнее. Критерием оценки эффективности выступает отношение производительности оборудования к величине его стоимости.

Классификация по типоразмерам. Персональные компьютеры можно классифицировать по типоразмерам. Так, различают ***настольные (desktop), портативные (notebook)*** и ***карманные (palmtop)*** модели.

Настольные модели распространены наиболее широко. Они являются принадлежностью рабочего места. Эти модели отличаются простотой изменения конфигурации за счет несложного подключения дополнительных внешних приборов или установки дополнительных внутренних компонентов Достаточные размеры корпуса в настольном исполнении позволяют выполнять большинство подобных работ без привлечения специалистов, а это позволяет настраивать компьютерную систему оптимально для решейия именно тех задач, для которых она была приобретена.

Портативные модели удобны для транспортировки. Их используют бизнесмены, коммерсанты, руководители предприятий и организаций, проводящие много времени в командировках и переездах. С портативным компьютером можно работать при отсутствии рабочего места. Особая привлекательность портативных компьютеров связана с тем, что их можно использовать в качестве средства связи. Подключив такой компьютер к телефонной сети, можно из любой географической точки установить обмен данными между ним и центральным компьютером своей организации. Так производят обмен данными, передачу приказов и распоряжений, получение коммерческих данных, докладов и отчетов. Для эксплуатации на рабочем месте портативные компьютеры не очень удобны, но их можно подключать к настольным компьютерам, используемым стационарно.

Карманные модели выполняют функции «интеллектуальных записных книжек». Они позволяют хранить оперативные данные и получать к ним быстрый доступ. Некоторые карманные модели имеют жестко встроенное программное обеспечение, что облегчает непосредственную работу, но снижает гибкость в выборе прикладных программ.

Мобильные вычислительные устройства сочетают в себе функции карманных моделей компьютеров и средств мобильной связи (сотовых радиотелефонов). Их отличительная особенность — возможность мобильной работы с Интернетом, а в ближайшем будущем и возможность приема телевизионных передач. Дополнительно *МВУ* комплектуют средствами связи по инфракрасному лучу, благодаря которым эти карманные устройства могут обмениваться данными с настольными ПК и друг с другом.

Классификация по совместимости. В мире существует множество различных видов и типов компьютеров. Они выпускаются разными производителями, собираются из разных деталей, работают с разными программами. При этом очень важным вопросом становится *совместимость* различных компьютеров между собой. От совместимости зависит

взаимозаменяемость узлов и приборов, предназначенных для разных компьютеров, возможность переноса программ с одного компьютера на другой и возможность совместной работы разных типов компьютеров с одними и теми же данными.

Аппаратная совместимость. По аппаратной совместимости различают так называемые *аппаратные платформы.* В области персональных компьютеров сегодня наиболее широко распространены две аппаратные платформы — *IBM PC* и *Apple Macintosh.* Кроме них существуют и другие платформы, распространенность которых ограничивается отдельными регионами или отдельными отраслями. Принадлежность компьютеров к одной аппаратной платформе повышает совместимость между ними, а принадлежность к разным платформам — понижает.

Кроме аппаратной совместимости существуют и другие виды совместимости: совместимость на уровне операционной системы, программная совместимость, совместимость на уровне данных.

Классификация по типу используемого процессора. *Процессор* — основной компонент любого компьютера. В электронно-вычислительных машинах это специальный блок, а в персональных компьютерах — специальная микросхема, которая выполняет все вычисления в компьютере. Даже если компьютеры принадлежат одной аппаратной платформе, они могут различаться по типу используемого процессора. Основные типы процессоров для платформы *IBM PC* мы рассмотрим в соответствующем разделе, а здесь укажем на то, что тип используемого процессора в значительной (хотя и не в полной) мере характеризует технические свойства компьютера.

15. Обшая структура компьютера.

Персональный компьютер — универсальная техническая система. Его *конфигурацию* (состав оборудования) можно гибко изменять по мере необходимости. Тем не менее, существует понятие *базовой конфигурации,* которую считают типовой. В таком комплекте компьютер обычно поставляется. Понятие базовой конфигурации может меняться. В настоящее время в базовой конфигурации рассматривают четыре устройства

- системный блок;
- монитор;
- клавиатура;
- мышь.

Системный блок представляет собой основной узел, внутри которого установлены наиболее важные компоненты. Устройства, находящиеся внутри системного блока, называют *внутренними,* а устройства, подключаемые к нему снаружи, — *внешними.* Внешние дополнительные устройства, предназначенные для ввода, вывода и длительного хранения данных, также называют *периферийными.*

По внешнему виду системные блоки различаются формой корпуса.

Корпуса персональных компьютеров выпускают в горизонтальном (*desktop)* и вертикальном *(tower)* исполнении. Корпуса, имеющие вертикальное исполнение, различают по габаритам: *полноразмерный (big tower), среднеразмерный (midi tower)* и *малоразмерный (mini tower).*

Базовая конфигурация компьютерной системы

Среди корпусов, имеющих горизонтальное исполнение, выделяют *плоские* и *особо плоские (slim).*

Кроме формы, для корпуса важен параметр, называемый *форм-фактором.* От него зависят требования к размещаемым устройствам. Прежним стандартом корпуса персональных компьютеров был форм-фактор *AT,* в настоящее время в основном используются корпуса форм-фактора *ATX.* Форм-фактор корпуса должен быть обязательно согласован с форм-фактором главной (системной) платы компьютера, так называемой *материнской платы* (см. ниже).

Корпуса персональных компьютеров поставляются вместе с блоком питания и, таким образом, мощность блока питания также является одним из параметров корпуса. Для массовых моделей достаточной является мощность блока питания 250-300 Вт.

Монитор — устройство визуального представления данных. Это не единственно возможное, но главное устройство вывода. Его основными потребительскими параметрами являются: тип, размер и шаг маски экрана, максимальная частота регенерации изображения, класс защиты.

Сейчас наиболее распространены мониторы двух основных типов на основе электронно-лучевой трубки (ЭЛТ) и плоские жидкокристаллические (ЖК). ЭЛТ-мониторы обеспечивают лучшее качество изображения, но в пользу жидкокристаллических мониторов говорит их компактность, небольшой вес, идеально плоская поверхность экрана.

Размер монитора измеряется между противоположными углами видимой части экрана по диагонали. Единица измерения — дюймы. Стандартные размеры: 14"; 15"; 17"; 19"; 20"; 21". В настоящее время наиболее универсальными являются мониторы размером 15 (ЖК) и 17 дюймов (ЭЛТ), а для операций с графикой желательны мониторы размером 19-21 дюйм (ЭЛТ).

Изображение на экране ЭЛТ-монитора получается в результате облучения люми- нофорного покрытия остронаправленным пучком электронов, разогнанных в вакуумной колбе. Для получения цветного изображения люминофорное покрытие имеет точки или полоски трех тйпов, светящиеся красным, зеленым и синм цветом.

Клавиатура — клавишное устройство управления персональным компьютером. Служит для ввода *алфавитно-цифровых (знаковых*) данных, а также команд управления. Комбинация монитора и клавиатуры обеспечивает простейший *интерфейс пользователя.* С помощью клавиатуры управляют компьютерной системой, а с помощью монитора получают от нее отклик.

Клавиатура относится к стандартным средствам персонального

компьютера. Ее основные функции не нуждаются в поддержке специальными системными программами (драйверами). Необходимое программное обеспечение для начала работы с компьютером уже имеется в микросхеме ПЗУ в составе базовой системы ввода-вывода *(BIOS),* и потому компьютер реагирует на нажатия клавиш сразу после включения.

Мышь — устройство управления манипуляторного типа. Представляет собой плоскую коробочку с двумя-тремя кнопками. Перемещение мыши по плоской поверхности синхронизировано с перемещением графического объекта *{указателя мыши)* на экране монитора.

В отличие от рассмотренной ранее клавиатуры мышь не является стандартным органом управления, и персональный компьютер не имеет для нее выделенного порта. Для мыши нет и постоянного выделенного прерывания, а базовые средства ввода и вывода *{BIOS)* компьютера, размещенные в постоянном запоминающем устройстве (ПЗУ), не содержат программных средств для обработки прерываний мыши.

Материнская плата — основная плата персонального компьютера. На ней размещаются:

- *процессор* — основная микросхема, выполняющая большинство математических и логических операций;
- *микропроцессорный комплект (*[u]*ипсет)* — набор микросхем, управляющих работой внутренних устройств компьютера и определяющих основные функциональные возможности материнской платы;
- *шины* — наборы проводников, по которым происходит обмен сигналами между внутренними устройствами компьютера;

оперативная память (оперативное запоминающее устройство, ОЗУ) — набор микросхем, предназначенных для временного хранения данных, когда компьютер включен; »

- *ПЗУ* (*постоянное запоминающее устройство)* — микросхема, предназначенная для длительного хранения данных, в том числе и когда компьютер выключен;
- разъемы для подключения дополнительных устройств *{слоты).*
- *Оперативная память {RAM — Random Access Memory)* — это массив кристаллических ячеек, способных хранить данные. Существует много различных типов оперативной памяти, но с точки зрения физического принципа действия различают *динамическую память {DRAM)* и *статическую память {SRAM).*
- *Ячейки динамической памяти {DRAM)* можно представить в виде микроконденсаторов, способных накапливать заряд на своих обкладках. Это наиболее распространенный и экономически доступный тип памяти. Недостатки этого типа связаны, во-первых, с тем, что как при заряде, так и при разряде конденсаторов неизбежны переходные процессы, то есть запись данных происходит сравнительно медленно. Второй важный недостаток связан с тем, что заряды ячеек имеют свойство рассеиваться в пространстве, причем весьма быстро. Если оперативную память постоянно не

«подзаряжать», утрата данных происходит через несколько сотых долей секунды. Для борьбы с этим явлением в компьютере происходит постоянная *регенерация*

• *{освежение, подзарядка)* ячеек оперативной памяти. Регенерация осуществляется несколько десятков раз в секунду и вызывает непроизводительный расход ресурсов вычислительной системы.

• Процессор — основная микросхема компьютера, в которой и производятся все вычисления. Конструктивно процессор состоит из ячеек, похожих на ячейки оперативной памяти, но в этих ячейках данные могут не только храниться, но и изменяться. Внутренние ячейки процессора называют *регистрами.* Важно также отметить, что данные, попавшие в некоторые регистры, рассматриваются не как данные, а как команды, управляющие обработкой данных в других регистрах. Среди регистров процессора есть и такие, которые в зависимости от своего содержания способны ■модифицировать исполнение команд. Таким образом, управляя засылкой данных в разные регистры процессора, можно управлять обработкой данных. На этом и основано исполнение программ.

• С остальными устройствами компьютера, и в первую очередь с оперативной памятью, процессор связан несколькими группами проводников, называемых *шинами.* Основных шин три: *шина данных, адресная шина* и *командная шина.*

Периферийные устройства персонального компьютера подключаются к его интерфейсам и предназначены для выполнения вспомогательных операций. Благодаря им компьютерная система приобретает гибкость и универсальность.

По назначению периферийные устройства можно подразделить на:

- устройства ввода данных;
- устройства вывода данных;
- устройства хранения данных;
- устройства обмена данными.

16. Структура персонального компьютера. Внутри-машинный системный интерфейс. Микро процессор и его функции.

Архитектурой называют прежде всего систему составляющих компьютер устройств и взаимосвязей между ними, а также совокупность правил, по которым происходит это взаимодействие.

Главными устройствами являются процессор и память. Именно взаимодействием этих компонентов определяется возможность компьютера производить вычисления. Линии связи, по которым данные передаются из процессора в память и обратно, называют ся *шиной.* Обычно это электрический провод (сейчас появились оптоволоконные провода). Линий связи в компьютере мною, и они выполняют множество разных функций. Принято делить линии связи всей шины на шину данных, шину

адреса, шипу управления и шину питания. Кроме того, процессор и память должны быть связаны проводами со многими другими устройс: вами компьютера. В современных компьютерах одна и та же шии;| используется для обмена данными как между процессором и па мятью, так и между процессором и всеми портами ввода-вывода Такая шина называется *общей шиной.*

Реально часть устройств подключается к об щей шине не непосредственно, а через одну из вспомогательных шин, которая, в свою очередь, присоединяется к общей шипе Такие шины называются *локальными шинами. Внутримашинным системным интерфейсом* называется вся си стема связей и сопряжений узлов и блоков компьютера между собой. Интерфейс включает совокупность электрических про но дов, электронных микросхем сопряжения с компонентами ком пьютера, соглашений о передаче и преобразовании сигналом Интерфейс с общей шиной называется односвязным.

Микропроцессор и его функции

Процессор состоит из огромного количества электронных микросхем, сосредоточенных в микроскопическом объеме. Быть может, процессор является самым сложным устройством в мире.

Регистры процессора, представляющие собой наиболее быстродействующую часть памяти компьютера, конструктивно расположены внутри процессора, и время доступа к данным в регистрах значительно меньше, чем к данным в оперативной памяти.

Центральный процессор (ЦП) — это устройство, которое выполняет обработку информации в соответствии с выполняемой компьютером программой, находящейся в оперативной памяти и

состоящей из отдельных команд, понятных для процессора. В каждой команде содержатся сведения о том, откуда взять исходные данные, какую операцию над ними выполнять и куда поместить результат. Процессор выполняет следующие функции:

- чтение команд из оперативной памяти и их дешифрация;
- чтение данных из оперативной памяти и портов ввода-вы вода;
- запись данных в оперативную память или их пересылка в порты ввода-вывода;
- прием и обработка запросов и команд от адаптеров внешних устройств;
- выработка управляющих сигналов для всех прочих устройств компьютера.

Функционально процессор состоит из двух компонентов -операционной и интерфейсной частей. Операционная часть включает устройство управления, арифметико-логическое устройство и процессорную память (регистры общего назначения — РОН). Интерфейсная часть включает микросхемы управления шиной и портами, а также адресный и командный регистры.

Устройство управления является наиболее сложной частью процессора. Оно вырабатывает сигналы, которые управляют все ми устройствами компьютера, и процессором в частности.

- Арифметико-логическое устройство (АЛУ) предназначено для выполнения арифметических и логических операций. Операнды операции перед этим должны быть размещены в регистрах общего назначения. Результат также помещается в регистр общего назначения.

17. Устройства вывода информации. Принтер.

Устройства вывода информации предназначены для предстаи-ления результатов работы компьютера в «человеческом» виде: кроме видеомонитора, о котором шла речь выше, это принтер, предназначенный для бумажной печати текстовой и графической информации, плоттер (или графопостроитель), предназначенный для печати графиков и чертежей, и звуковые колонки.

Управление работой большинства устройств ввода-вывода компьютера осуществляется при помощи портов. Напомним, что портом называется виртуальная ячейка, соответствующая внешнему входу (или выходу) в компьютере.

Принтер является основным средством бумажного вывода.

Плоттер — это фактически большой принтер, предназначен ный для построения чертежей. Он ориентирован на работу со специальными программами, например при печати чертежей и системах проектирования (системах автоматизации проектиропл ния — САПР). Принципы его действия те же, что и у принтера.

Монитор (дисплей) компьютера предназначен для вывода текстовой и графической информации. Мониторы персональных компыо теров могут работать в двух режимах — текстовом или графическом. В графическом режиме экран состоит из точек, полученных разбиением экрана на большое количество строк и столбцов. Эти точки называются пикселами. Количество пикселов на экране называется разрешающей способностью монитора в данном режиме. Сейчас мониторы персональных компьютеров могут работать в режимах 480 х 640, 600 х 800, 768 х 1024, 864 х 1152 или 1024 х 1280 пикселов. В графическом режиме экран состоит из точек, полученных рачбиением экрана на большое количество строк и столбцов. Количество пикселов на экране называется разрешающей способностью монитора. Сейчас мониторы персональных компьютеров могут работать в режимах 480 х 640, 600 х 800, 768 х 1024, 864 х 1152 или 1024 х 1280 пикселов.

Принтер.

Принтер является основным средством бумажного вывода. Принтеры бывают последовательные, строчные и страничные. По принципу действия

различают принтеры ударного и безударного действия. По способу печати принтеры делятся на матричные и символьные.

Матричные принтеры ударного действия содержат вертикальный ряд (иногда два ряда) игл или молоточков, которые вколачи-иают краситель с ленты прямо в бумагу, формируя последовательно символ за символом. Игольчатые принтеры имеют приемлемое качество печати, невысокую цену самого устройства, а также расходных материалов и бумаги. Для этих принтеров обычно возможно использование как форматной, так и рулонной бумаги. Головка принтера может быть оснащена 9, 18 или 24 иголками.

Более высокую производительность обеспечивают построчные (постраничные) матричные принтеры. Вместо маленьких точечно-матричных головок они используют длинные массивы с большим количеством игл, при этом достигается скорость порядка 1500 строк в минуту. Матричные ударные печатающие устройства создают много шума, что является существенным недостатком.

Струйные принтеры относятся к безударным печатающим устройствам и работают практически бесшумно. Струйные чернильные принтеры относятся к классу последовательных матричных безударных печатающих устройств. Они же, в свою очередь, подразделяются на устройства непрерывного и дискретного действия. 11оследние могут использовать либо пузырьковую технологию, либо ш.езоэффект. При печати высокого качества скорость вывода не превосходит обычно 2—3 страниц в минуту. В лазерных принтерах используется электрографический сип соб создания изображения — примерно такой же, как и в ксерок сах. Кроме лазерных существуют LED-принтеры, которые полу чили свое название из-за того, что полупроводниковый лазер и них был заменен «гребенкой» мельчайших светодиодов.

18. Понятия алгоритма.

Во всех сферах своей деятельности, в частности в сфере обработки информации, человек сталкивается с различными способами или методиками решения разнообразных задач. Они определяют порядок выполнения действий для получения желаемого результата — мы можем трактовать это как первоначальное или интуитивное определение алгоритма.

Термин ***алгоритм*** происходит от имени средневекового узбекского математика Аль-Хорезми, который еще в 825 году описал правила выполнения четырех арифметических действий в десятичной системе счисления. Процесс их выполнения и был назван ***алгоризмом.***

Впоследствии вместо слова алгоризм стали употреблять латинизированное слово ***алгорисмуСу*** смысл которого состоял в комбинировании четырех операций арифметического исчисления: сложения, вычитания, умножения и деления. Затем алгоритмус преобразовался в ***алгорифм***. Смысл этого понятия чаще всего связывался с алгорифмами Евклида — описаниями процессов нахождения наибольшего общего

делителя двух натуральных числе, наибольшей общей меры двух отрезков и т. п.

Под алгоритмом же понимали конечную последовательность точно сформулированных правил, которые позволяют решать различные классы задач. Такое определение алгоритма не является строго математическим, так как в нем не содержится точной характеристики того, что следует понимать под классами задач и под правилами их решения. Таким образом, можно нестрого определить алгоритм как однозначно трактуемую процедуру решения задачи.

Алгоритм- это система однозначных инструкций (указаний), которая определяет последовательность действий над выбранными обьектами с целью полуцения результата за конечное число шагов.

Дополнительные требования о выполнении алгоритма за конечное время для любых входных данных приводят к еще одному неформальному определению алгоритма.

Алгоритм- это заданное на некотором языке конечное предписание,задающее конечную последовательность выполнимых и точно определенных элементарных операций для решиния задачи.

Пусть ДД — область (множество) исходных данных задачи Да *Л* — множество возможных результатов, тогда мы можем говорить, что алгоритм осуществляет отображение -> /?.

Поскольку такое отображение может быть не полным, в теории алгоритмов вводятся понятия частичного и полного алгоритма. Алгоритм называется ***частичным***, если мы получаем результат только для некоторых значений Д принадлежащих ДД и ***полным***, если алгоритм получает правильный результат для всех значений ***ф*** принадлежащих *И!*.

Сегодня отсутствует одно исчерпывающе строгое определение понятия «алгоритм». Из разнообразных вариантов словесного определения наиболее удачные, видимо, принадлежат российским ученым А. Н. Колмогорову и А. А. Маркову.

Алгоритм (по Колмогорову) - это система вычислений,выполняемых по строво определенным правилам, которая после какого- либо числа шагов заведомо приводит к решени поставленной задачи.

Алгоритм (по Маркову) - это точное предписание, определяющее вычислительный прочесс, идуший от варьируемых исходных данных к искомомы результаты.

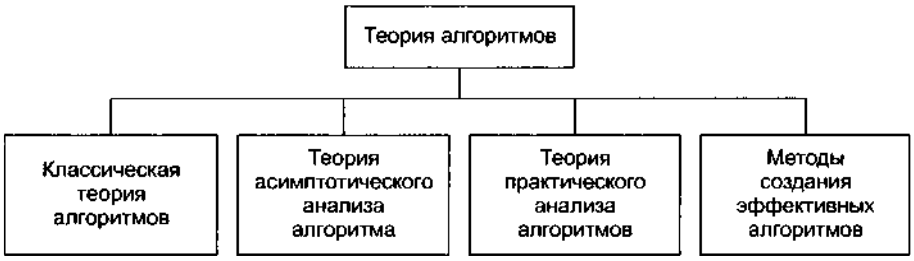
Алгоритмы можно подразделить на два класса: численные и логические. *Численные алгоритмы* — это алгоритмы, в соответствии с которыми решение задач сводится к арифметическим действиям.

Логические алгоритмы — это алгоритмы, в соответствии с которыми решение задач сводится к логическим действиям.

Несмотря на различие в определениях, к алгоритмам предъявляется ряд общих требований. Итак, алгоритм должен:

- содержать конечное количество элементарно выполнимых предписаний, то есть удовлетворять требованию конечности записи;

- выполнять конечное количество шагов при решении задачи, то есть удовлетворять требованию конечности действий;
- быть единым для всех допустимых исходных данных, то есть удовлетворять требованию универсальности;
- приводить к правильному по отношению к поставленной задаче решению, то есть удовлетворять требованию правильности.



19. Разветвляющие алгоритмы.

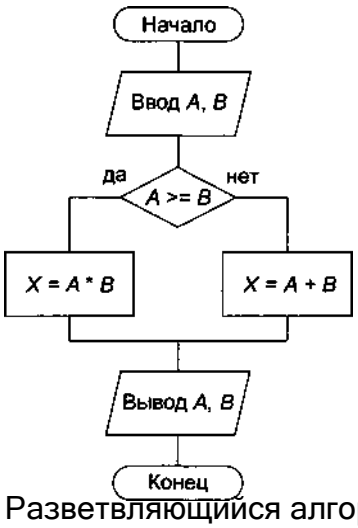
Для выполнения типичных последовательностей действий в алгоритме разработаны базовые алгоритмические конструкции в виде определенного набора блоков и стандартных средств их соединения. К базовым алгоритмическим конструкциям относятся ***линейные***, ***разветвляющиеся*** и ***циклические.***

В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в разветвляющиеся алгоритмы входит условие, в зависимости от выполнения или невыполнения которого выполняется та или иная последовательность команд (действий). Пример ветвляющегося алгоритма приведен на

Разветвляющие алгоритм - это алгоритм в котором действия выполняются по одйой из| возможных ветвей решения заданий в зависимости от выполнения условий.

В качестве условия в разветвляющемся алгоритме может быть использовано любое понятное исполнителю утверждение, которое может соблюдаться (быть истинным) или не соблюдаться (быть ложным). Такое утверждение может быть выражено как словами, так и формулой. Таким образом, алгоритм ветвления состоит из условия и двух последовательностей команд.

Многократные повторения одних и тех же действий обеспечиваются при помощи циклических алгоритмов .В цикл входят в качестве составляющих элементов блок проверки условия и блок, называемый телом цикла. Перед началом цикла осуществляются операции присваивания начальных значений тем объектам, которые используются в теле цикла.

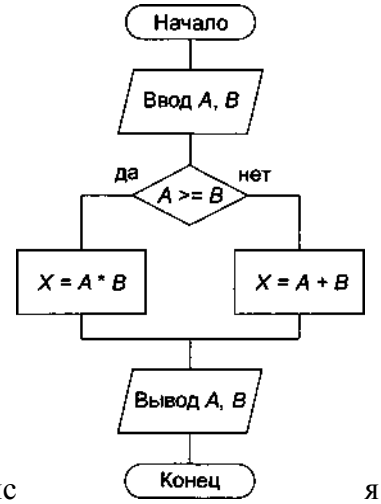


Разветвляющийся алгоритм

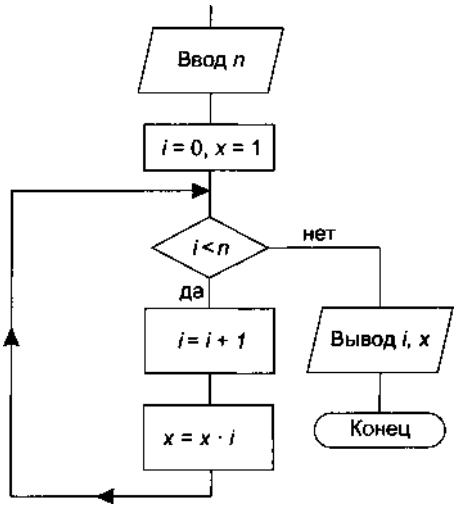
20. Циклические алгоритмы.

Для выполнения типичных последовательностей действий в алгоритме разработаны базовые алгоритмические конструкции в виде определенного набора блоков и стандартных средств их соединения. К базовым алгоритмическим конструкциям относятся ***линейные***, ***разветвляющиеся*** и ***циклические.***

Многократные повторения одних и тех же действий обеспечиваются при помощи циклических алгоритмов .В цикл входят в качестве составляющих элементов блок проверки условия и блок, называемый телом цикла. Перед началом цикла осуществляются операции присваивания начальных значений тем объектам, которые используются в теле цикла.



Разветвляющийс я алгоритм



Алгоритм с циклическим элементом

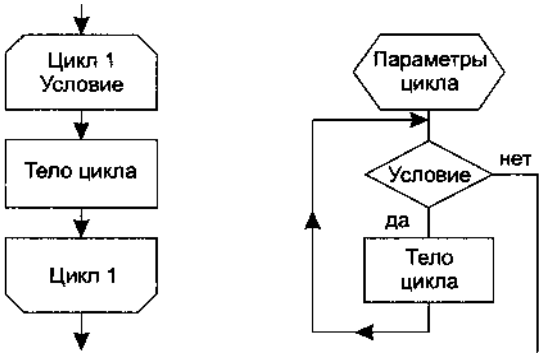
Циклический алгоритм - это алгоритм в котором некоторая часть операций (тело цикла) выполняется многократно.

Слово «многократно» в определении циклического алгоритма не значит «до бесконечности». Организация циклов, не приводящих к остановке в выполнении алгоритма, является нарушением требования его результативности — получения результата за конечное число шагов.

Если тело цикла расположено после проверки условий (цикл с предусловием), то при определенных условиях тело цикла не выполнится ни разу. Такой вариант организации цикла, управляемый предусловием, называется *циклом с предусловием* .

а) Общее обозначение по ГОСТ 19.701-90

б) Реализация

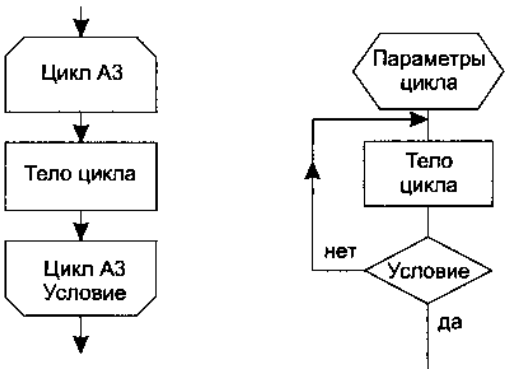


Цикл с предусловием

Другой вариант построения цикла состоит в том, что тело цикла выполняется по крайней мере один раз, и будет повторяться до тех пор, пока условие не станет истинным. Такая организация цикла, когда его тело расположено перед проверкой условия, носит название *цикла с постусловием*. Истинность условия в этом случае является условием выхода из цикла.

а) Общее обозначение по ГОСТ 19.701-90

б) Реализация



Цикл с постусловием

Таким образом, цикл с предусловием завершается, когда условие ложно, а цикл с постусловием — когда условие становится истинным.

Третий тип циклов, в которых тело цикла ***выполняется заданное число раз*** (то есть еще до начала выполнения цикла точно известно, сколько раз он будет выполнен), реализуется с помощью счетчика. В цикле со счетчиком значения счетчика в теле цикла итеративно увеличиваются.

Процесс решения сложной задачи довольно часто сводится к решению нескольких более простых задач методом последовательной детализации. Соответственно, процесс разработки сложного алгоритма может разбиваться на этапы составления отдельных алгоритмов, которые называются вспомогательными. Каждый такой вспомогательный алгоритм описывает решение какой-либо подзадачи.

Процесс построения алгоритма методом последовательной детализации состоит в следующем. Сначала алгоритм формулируется в «обобщающих» блоках, которые могут быть непонятны исполнителю и записываются, как вызовы вспомогательных алгоритмов. Затем происходит детализация, и все вспомогательные алгоритмы детализируются до уровня команд, понятных исполнителю.

21. Языки программирования, классификация

Компьютерные программы создают ***программисты*** — люди, обученные процессу их составления (***программированию***). Мы знаем, что программа — это логически упорядоченная последовательность команд, необходимых для управления компьютером (выполнения им конкретных операций), поэтому программирование сводится к созданию последовательности команд, необходимой для решения определенной задачи.

Самому написать программу в машинном коде весьма сложно, причем эта сложность резко возрастает с увеличением размера программы и трудоемкости решения нужной задачи. Условно можно считать, что машинный код приемлем, если размер программы не превышает нескольких десятков байтов и нет потребности в операциях ручного ввода/вывода данных.

Поэтому сегодня практически все программы создаются с помощью языков программирования. Теоретически программу можно написать и

средствами обычного человеческого (естественного) языка — это называется программированием на *метаязыке* (подобный подход обычно используется на этапе составления алгоритма), но автоматически перевести такую программу в машинный код пока невозможно из-за высокой неоднозначности естественного языка.

Языки программирования — искусственные языки. От естественных они отличаются ограниченным числом «слов», значение которых понятно транслятору, и очень строгими правилами записи команд (*операторов*). Совокупность подобных требований образует *синтаксис* языка программирования, а *смысл* каждой команды и других конструкций языка — его *семантику.* Нарушение формы записи программы приводит к тому, что транслятор не может понять назначение оператора и выдает сообщение о синтаксической ошибке, а правильно написанное, но не отвечающее алгоритму использование команд языка приводит к семантическим ошибкам (называемым еще логическими ошибками или ошибками времени выполнения).

Процесс поиска ошибок в программе называется *тестированием,* процесс устранения ошибок — *отладкой.*

Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется *языком программирования низкого уровня.* В данном случае «низкий уровень» не значит «плохой». Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора.

Языком самого низкого уровня является *язык ассемблера,* который просто представляет каждую команду машинного кода, но не в виде чисел, а с помощью символьных условных обозначений, называемых *мнемониками.* Однозначное преобразование одной машинной инструкции в одну команду ассемблера называется *транслитерацией.* Так как наборы инструкций для каждого модели процессора отличаются, конкретной компьютерной архитектуре соответствует свой язык ассемблера, и написанная на нем программа может быть использована только в этой среде.

С помощью языков низкого уровня создаются очень эффективные и компактные программы, так как разработчик получает доступ ко всем возможностям процессора. С другой стороны, при этом требуется очень хорошо понимать устройство компьютера, затрудняется отладка больших приложений, а результирующая программа не может быть перенесена на компьютер с другим типом процессора. Подобные языки обычно применяют для написания небольших системных приложений, драйверов устройств, модулей стыковки с нестандартным оборудованием, когда важнейшими требованиями становятся компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам. В некоторых областях, например в машинной графике, на языке ассемблера пишутся библиотеки, эффективно реализующие требующие интенсивных вычислений алгоритмы обработки изображений.

Языки программирования высокого уровня значительно ближе и понятнее человеку, нежели компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан транслятор этого языка. Разрабатывать программы на языках высокого уровня с помощью понятных и мощных команд значительно проще, а ошибок при создании программ допускается гораздо меньше.

Языки программирования принято делить на пять ***поколений.*** В первое поколение входят языки, созданные в начале 50-х годов, когда первые компьютеры только появились на свет. Это был первый язык ассемблера, созданный по принципу «одна инструкция — одна строка».

Расцвет второго поколения языков программирования пришелся на конец 50-х — начало 60-х годов. Тогда был разработан символический ассемблер, в котором появилось понятие переменной. Он стал первым полноценным языком программирования. Благодаря его возникновению заметно возросли скорость разработки и надежность программ.

Появление третьего поколения языков программирования принято относить к 60-м годам. В это время родились универсальные языки высокого уровня, с их помощью удается решать задачи из любых областей. Такие качества новых языков, как относительная простота, независимость от конкретного компьютера и возможность использования мощных синтаксических конструкций, позволили резко повысить производительность труда программистов. Понятная большинству пользователей структура этих языков привлекла к написанию небольших программ (как правило, инженерного или экономического характера) значительное число специалистов из некомпьютерных областей. Подавляющее большинство языков этого поколения успешно применяется и сегодня.

С начала 70-х годов по настоящее время продолжается период языков четвертого поколения. Эти языки предназначены для реализации крупных проектов, повышения их надежности и скорости создания. Они обычно ориентированы на специализированные области применения, где хороших результатов можно добиться, используя не универсальные, а проблемно-ориентированные языки, оперирующие конкретными понятиями узкой предметной области. Как правило, в эти языки встраиваются мощные операторы, позволяющие одной строкой описать такую функциональность, для реализации которой на языках младших поколений потребовались бы тысячи строк исходного кода.

Рождение языков пятого поколения произошло в середине 90-х годов. К ним относятся также системы автоматического создания прикладных программ с помощью визуальных средств разработки, без знания программирования. Главная идея, которая закладывается в эти языки, — возможность автоматического формирования результирующего текста на универсальных языках программирования (который потом требуется откомпилировать). Инструкции же вводятся в компьютер в максимально наглядном виде с помощью методов, наиболее удобных для человека, не

знакомого с программированием.

22. Программы трансиляторы

С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм. Чтобы получить работающую программу, надо этот текст либо автоматически перевести в машинный код (для этого служат программы***-компиляторы)*** и затем использовать отдельно от исходного текста, либо сразу выполнять команды языка, указанные в тексте программы (этим занимаются программы***-интерпретаторы).***

Интерпретатор берет очередной оператор языка из текста программы, анализирует его структуру и затем сразу исполняет (обычно после анализа оператор транслируется в некоторое промежуточное представление или даже машинный код для более эффективного дальнейшего исполнения). Только после того, как текущий оператор успешно выполнен, интерпретатор перейдет к следующему. При этом, если один и тот же оператор должен выполняться в программе многократно, интерпретатор всякий раз будет выполнять его так, как будто встретил впервые. Вследствие этого, программы, в которых требуется осуществить большой объем повторяющихся вычислений, могут работать медленно. Кроме того, для выполнения такой программы на другом компьютере там также должен быть установлен интерпретатор — ведь без него текст программы является просто набором символов.

По-другому можно сказать, что интерпретатор моделирует некую виртуальную вычислительную машину, для которой базовыми инструкциями служат не элементарные команды процессора, а операторы языка программирования.

Компиляторы полностью обрабатывают весь текст программы (он иногда называется ***исходный код).*** Они просматривают его в поисках синтаксических ошибок (иногда несколько раз), выполняют определенный смысловой анализ и затем автоматически переводят (***транслируют)*** на машинный язык — генерируют машинный код. Нередко при этом выполняется ***оптимизация*** с помощью набора методов, позволяющих повысить быстродействие программы (например, с помощью инструкций, ориентированных на конкретный процессор, путем исключения ненужных команд, промежуточных вычислений и т. д.). В результате законченная программа получается компактной и эффективной, работает в сотни раз быстрее программы, выполняемой с помощью интерпретатора, и может быть перенесена на другие компьютеры с процессором, поддерживающим соответствующий машинный код.

Основной недостаток компиляторов — трудоемкость трансляции языков программирования, ориентированных на обработку данных сложной структуры, часто заранее неизвестной или динамически меняющейся во

время работы программы. Тогда в машинный код приходится вставлять множество дополнительных проверок, анализировать наличие ресурсов операционной системы, динамически их захватывать и освобождать, формировать и обрабатывать в памяти компьютера сложные объекты, что на уровне жестко заданных машинных инструкций осуществить довольно трудно, а для ряда задач практически невозможно.

С помощью интерпретатора, наоборот, допустимо в любой момент остановить работу программы, исследовать содержимое памяти, организовать диалог с пользователем, выполнить сколь угодно сложные преобразования данных и при этом постоянно контролировать состояние окружающей программно-аппаратной среды, благодаря чему достигается высокая надежность работы. Интерпретатор при выполнении каждого оператора проверяет множество характеристик операцион- ной системы и при необходимости максимально подробно информирует разработчика о возникающих проблемах. Кроме того, интерпретатор очень удобен для использования в качестве инструмента изучения программирования, так как позволяет понять принципы работы любого отдельного оператора языка.

В реальных системах программирования перемешаны технологии и компиляции, и интерпретации. В процессе отладки программа может выполняться по шагам, а результирующий код не обязательно будет машинным — он даже может быть исходным кодом, написанным на другом языке программирования (это существенно упрощает процесс трансляции, но требует компилятора для конечного языка), или промежуточным машинно-независимым кодом абстрактного процессора, который в различных компьютерных архитектурах станет выполняться с помощью интерпретатора или компилироваться в соответствующий машинный код.

23. Объектно - ориентированные языки

Эта группа языков отличается от алгоритмических языков прежде всего решаемыми задачами. База данных — это файл (или группа файлов), представляющий собой упорядоченный набор ***записей,*** имеющих единообразную структуру и организованных по единому шаблону (как правило, в табличном виде). База данных может состоять из нескольких таблиц. Удобно хранить в базах данных различные сведения из справочников, картотек, журналов бухгалтерского учета и т. д.

При работе с базами данных чаще всего требуется выполнять следующие операции:

- создание/модификация свойств/удаление таблиц в базе данных;
- поиск, отбор, сортировка информации по запросам пользователей;
- добавление новых записей;
- модификация существующих записей;
- удаление существующих записей.

Первые базы данных появились очень давно, как только появилась потребность в обработке больших массивов информации и выборки групп записей по определенным признакам. Для этого был создан ***структурированный язык запросов SQL (,Structured Query Language).*** Он основан на мощной математической теории и позволяет выполнять эффективную обработку баз данных, манипулируя не отдельными записями, а ***группами*** записей.

Для управления большими базами данных и их эффективной обработки разработаны СУБД (Системы Управления Базами Данных). Практически в каждой СУБД помимо

поддержки языка ***SQL*** имеется также свой уникальный язык, ориентированный на особенности этой СУБД и не переносимый на другие системы. Сегодня в мире насчитывается три ведущих производителя СУБД: ***Microsoft (SQL Server), IBM (DB2)*** и ***Oracle.*** Их продукты нацелены на поддержку одновременной работы тысяч пользователей в сети, а базы данных могут храниться в распределенном виде на нескольких серверах. В каждой из этих СУБД реализован собственный диалект ***SQL,*** ориентированный на особенности конкретного сервера, поэтому ***SQL***-программы, подготовленные для разных СУБД, друг с другом, как правило, несовместимы.

С появлением персональных компьютеров были созданы так называемые настольные СУБД. Родоначальником современных языков программирования баз данных для ПК принято считать СУБД ***dBase II,*** язык которой был интерпретируемым. Затем для него были созданы компиляторы, появились СУБД ***FoxPro*** и ***Clipper,*** поддерживающие диалекты этого языка. Сегодня самой распространенной настольной СУБД стала система ***Microsoft Access.***

С активным развитием глобальной сети было создано немало реализаций популярных языков программирования, адаптированных специально для Интернета. Все они отличаются характерными особенностями: языки являются интерпретируемыми, интерпретаторы для них распространяются бесплатно, а сами программы — в исходных текстах. Такие языки называют ***скрипт-языками.***

HTML. Общеизвестный язык для оформления документов. Он очень прост и содержит элементарные команды форматирования текста, добавления рисунков, задания шрифтов и цветов, организации ссылок и таблиц. Все Weè-страницы написаны на языке ***HTML*** или используют его расширения.

Perl. В 80-х годах Ларри Уолл разработал язык ***Perl.*** Он задумывался как средство эффективной обработки больших текстовых файлов, генерации текстовых отчетов и управления задачами. По мощности ***Perl*** значительно превосходит языки типа Си. В него введено много часто используемых функций работы со строками, массивами, всевозможные средства преобразования данных, управления процессами, работы с системной информацией и др.

PHP. Расмус Лердорф, активно использовавший Рег/-скрипты, в 1995 году решил улучшить этот язык, упростив его и дополнив встроенными

средствами доступа к базам данных. В результате появилась разработка ***Personal Contents Page/Forms Interpreter (PHP/FI).*** Уже через пару лет программы на ее основе использовались на 50 тыс. сайтов. В 1997 году ее значительно усовершенствовали Энди Гутмане и
Зив Сураски, и под названием ***PHP*** 3.0 этот язык быстро завоевал популярность у создателей динамических сайтов во всем мире.
Tcl/Tk. В конце 80-х годов Джон Аустираут придумал популярный скрипт-язык ***Tel*** и библиотеку ***Tk.*** В ***Tel*** он попытался воплотить видение идеального скрипт- языка. Язык ***Tel*** ориентирован на автоматизацию рутинных процессов и состоит из мощных команд, предназначенных для работы с абстрактными нетипизирован- ными объектами. Он независим от типа системы и при этом позволяет создавать программы с графическим интерфейсом.
VRML. В 1994 году был создан язык ***VRML*** для организации виртуальных трехмерных интерфейсов в Интернете. Он позволяет описывать в текстовом виде различные трехмерные сцены, освещение и тени, текстуры (покрытия объектов), создавать свои миры, путешествовать по ним, «облетать» со всех сторон, вращать в любых направлениях, масштабировать, регулировать освещенность и т. д.
XML. В августе 1996 года WWW-консорциум, ответственный за стандарты на Интернет-технологии, приступил к подготовке универсального языка разметки структуры документов, базировавшегося на достаточно давно созданной в ***IBM*** технологии ***SGML.*** Новый язык получил название ***XML.*** Сегодня он служит основой множества системных, сетевых и прикладных приложений, позволяя представлять в прозрачном для пользователей и программ текстовом виде различные аспекты внутренней структуры иерархически организованных документов. В недалеком будущем он может стать заменой ***HTML.***

24. Основные элементы языка Паскаль

Как уже было сказано в прошлой главе, язык программирования высокого уровня 11 аскаль[1] был разработан швейцарским профессором Никлаусом Виртом для обучения программированию как систематической дисциплине, а также как надежный и эффективный язык разработки программ для тех компьютеров, которые не пользовались во время создания языка, то есть в конце 1960-х годов. Язык окавшей настолько удачным, что вышел за рамки этих задач, быстро завоевал популярность и широко используется до сих пор.
Международным признанием языка Паскаль можно считать выпуск

стандартов, описывающих синтаксис этого языка. Международные стандарты в мире програм-мирования играют роль своеобразных кодексов поведения. Если программист придерживается стандартов, его программы оказываются *переносимыми,* то есть могут выполняться на разных компьютерах и в разных операционных системах. Это хорошее свойство программ, и осознать это несложно. Представьте себе человека, нарушающего принятые в обществе законы поведения. Такому человеку сложно будет ужиться с окружающими, жизнь его будет трудной. С другой стороны, нередки случаи, когда при разработке конкретных реализаций языка и систем программирования допускается нарушение стандартов, обычно за счет введения дополнительных возможностей, облегчающих жизнь программиста. Примером такого расширения является система программирования Turbo Pascal фирмы Borland (сейчас Inprise). Современный язык Паскаль, точнее его синтаксис, описывается в специальных документах — международных стандартах ISO/IEC 7185:1990(E) (Pascal) и ISO/ IEC 10206:1990(E) (Extended Pascal или «расширенный Паскаль»), Эти документы основаны на предыдущих попытках выработать стандарт языка и созданы рабочими группами, состоящими из большого числа специалистов в области программирования.

Стандарты ISO/IEC 7185 и 10206 не определяют допустимые размеры или сложность программ, написанных на языке Паскаль, методы трансляции программ в машинные коды, особенности архитектуры компьютера для выполнения программ на Паскале или его операционную систему, сообщения об ошибках, методы типографской записи программ и некоторые другие вопросы Язык Турбо Паскаль состоит приблизительно из 80 зарезервированных слов и специальных символов. Алфавит языка составляют буквы латинского алфавита, как строчные, так и прописные, цифры, а также специальные символы, такие как, напри- мер, +, -, _. Специальными символами языка являются и некоторые *пары* символов. Как уже отмечалось, зарезервированные слова в языке Паскаль могут применяться только по своему прямому назначению, то есть в качестве имен операторов, названий операций и т. д. Алфавитный список зарезервированных слов приведен в табл. 2.1. В большинстве случаев овладение даже небольшой частью этого «словаря» достаточно для начала успешной работы по программированию на Паскале. В дальнейшем мы разберем применение приведенных в данной таблице зарезервированных слов, а сейчас лишь кратко поясним смысл некоторых из них, наиболее часто используемых в программах на Паскале.

- Заголовки, то есть первые операторы программ и библиотечных модулей, — это program и unit.
- Для описания переменных, констант и составных частей программы — подпрограмм-процедур и подпрограмм-функций — используются зарезервированные слова var, const, procedure и function.
- Операторы описания типов переменных, задаваемых пользователем, — это type, array, string, record...end, file of.

Слова, используемые для программирования составных операторов, а также начинающие и оканчивающие последовательность исполняемых операторов программы, begin и end.

Алфавит, специальные символы и специальные слова языка Паскаль

- Операторами, управляющими ходом выполнения программы (они так и на зываются — *управляющие операторы),* являются if...then...else, for...to...do, repeat...unti 1, case...of...end, for...downto...do, whi 1 e...do.
- В библиотечных модулях используются зарезервированные слова i mpl ementati on, interface.
- Зарезервированные слова для обозначения арифметических и логических оно раций — di v, mod, shl, shr, and, or, not и некоторые другие.
- U В программах, написанных с использованием методов объектно-ориентирова]ню го программирования, применяются зарезервированные слова object, constructor, destructor, public и virtual.

Зарезервированные слова языка Паскаль

absolute	end	near	shr
and	external	nil	string
array	far	not	then
assemble	file	object	to
begin	for	of	type
break	function	or	unit
case	goto	packed	until
const	if	private	uses
construct	implementatio	procedure	var
continue	in	program	virtual
dostructor	inline	public	while
div	interface	record	with
do	interrupt	repeat	xor
down to	label	set	
also	mod	shl	

Кроме перечисленных зарезервированных слов есть ключевые слова, которые нс пользуются для обозначения типов переменных или других объектов програм мы. Забегая вперед, приведу пример программы), показывающий, что в определенных ситуациях допускается использование ключевых слов в ка честве имен переменных (переменные Real и Integer). Использовать так зарезер иированные слова категорически запрещено! Да и применение
ключевых слои по по назначению считается очень плохим стилем программирования. Его еде дует избегать

Листинг

```
Real := 1:
Integer := 5:
WriteLn(Real);
WriteLn(Integer); end.
```

В программах на языке Паскаль используются как отдельные специальные символы, так и пары символов, которые имеют специальное значение. Перечень таких символов приведен в табл. 2.2.

Одиночные и двойные специальные символы языка Паскаль

Присваивание переменной, стоящей слева от символа, значения выражения, стоящего справа от символа

Разделитель операторов в программе ( )Скобки для арифметических и логических выражений

Разделитель в описаниях переменных и формате операторов вывода Многоточие для списков

| Бинарные операции (не только арифметические!)

Тоже

* То же

/ То же

Логическое равенство, элемент описания констант и типов Логическое неравенство Отношение «меньше чем»

Отношение «больше чем»

Отношение «меньше или равно»

>- Отношение «больше или равно»

Конец программы или модуля, а также десятичная точка в буквальных константах вещественного типа

Ограничители константы строкового типа (} Пары скобок для комментариев

(**) Тоже

Разделитель элементов списка

|| Скобки для ссылки на элемент массива или указания диапазона значений индекса

* Символ разыменования указателя

@ Операция получения указателя

6)@ Операция получения адреса

Символ подчеркивания (иногда заменяет пробел)

Пробел _____ _ _____

25. Идентификаторы, типы данных

Каждый шаг алгоритма суммирования сформулирован в общем виде. Компьютер не сможет выполнить этот алгоритм, поскольку из вышеприведенной записи непонятно, как выполняются считывание и вывод значений, а также как выполняется суммирование. Позже в этой главе мы убедимся, что сложение для компьютера — не такая простая операция, как может показаться.

Для каждого из шагов можно составить собственный алгоритм. Шаги этих алгоритмов будут содержать более детальные описания. Поэтому говорят,

что алгоритм имеет *иерархическую структуру.* Степень детализации (подробности описания) может оказаться недостаточной для реализации с помощью машинных команд. Тогда процесс *пошаговой детализации алгоритма* может быть продолжен. Степень детализации зависит от разных обстоятельств, в том числе и от того языка программирования, который предполагается использовать для записи алгоритма. При программировании на языках низкого уровня требуется подробное описание действий, а языки высокого уровня такой подробной детализации не требуют.

Основными объектами программирования являются *переменные.* Переменные в программе отличаются от переменных, используемых в записи математических формул. Несмотря на сходство терминов, правила использования переменных в программах для компьютера отличаются от правил работы с математическими переменными. Это различие необходимо понять и усвоить. В программировании переменную можно трактовать как одну или несколько физических ячеек оперативной памяти компьютера, которым присвоено определенное имя. Содержимое этих ячеек может изменяться, но имя переменной остается неизменным. В математике значение переменной в рамках определенной задачи неизменно, но меняется в других задачах из данного класса. Именно поэтому конструкция a := a + 1;

воспринимается программистом совершенно естественно, а уравнение $a = a + 1$ математик сочтет неверным. В первом случае имеется в виду вычисление суммы содержимого ячейки a и числовой константы 1 с последующим занесением полученного результата в ту же ячейку a. Второй случай равносилен неверному тождеству 0 = 1. Подробнее о переменных и их свойствах мы поговорим далее в этой главе.

(-оставим алгоритм решения следующей задачи. Пусть заданы два значения: *x* и *y.*

11еобходимо напечатать имя большей переменной. Для решения этой задачи достаточно сравнить оба значения и в зависимости от результата сравнения вывести па печать символ <<x» или символ «y». Таким образом, алгоритм будет выглядеть так.

1. Ввести значение *x.*
2. Ввести значение *y.*

Л. Если $x < y$, напечатать «y», иначе напечатать «x».

В этом алгоритме используются две *алгоритмические структуры — линейная последовательность операции* (шаги 1, 2) и *ветвление* (шаг 3, условный оператор).

Алгоритмические структуры описывают последовательность выполнения шагов алгоритма. В линейной последовательности шаги выполняются последовательно, один за другим, без повторений и без вариантов. Алгоритмическая структура «ветвление» называется так потому, что после передачи ей управления выполнение алгоритма может пойти по одной из двух (в общем случае — из нескольких) возможных ветвей. То, какая ветвь будет выбрана, зависит от выполнения условия. Л инейная последовательность в данном примере состоит из блоков *ввода-вывода*

данных.

Рассмотрим еще один пример составления алгоритма. Пусть необходимо вычислить целую степень числа x^n для произвольных значений степени и основания, используя только операцию умножения. Проблема здесь заключается в том, что конкретное значение степени заранее не известно, следовательно, не известно и необходимое количество умножений. Для того чтобы решить эту проблему, введем в алгоритм вспомогательную переменную — счетчик числа выполненных умножений, а степень будем вычислять по рекуррентной формуле:

$z_0 = 1$, $z_n = z_{n-x} \times x$.

Запись алгоритма вычисления произвольной целочисленной степени числа приводится ниже.

1. Ввести значения x и н .
2. Присвоить z_0 начальное значение 1.
3. Присвоить вспомогательной переменной *i* начальное значение 0.
4. Присвоить2результат выполнения операции z x x .
5. Присвоить *i* значение суммы *i* +1.

(i. Если $i < n$, перейти к шагу 4, иначе — остановить работу программы.

Этот алгоритм содержит ввод данных (шаг 1), *блок вычислений* (шаги 2,3), ветвле- цне (шаг 6), а также еще одну алгоритмическую структуру — *цикл* (шаги 4, 5 и 6).

11,11 кл представляет собой многократно повторяющуюся последовательность операторов и играет в программировании важнейшую роль. Кроме уже перечисленных структур иногда выделяют еще одну — *обход,* который представляет собой передачу управления, минуя несколько шагов алгоритма.

Для записи алгоритмов мы пользовались естественным языком. Иногда используют полуформальный язык с ограниченным словарем (часто на основе английского языка) — как промежуточный между естественным языком и языком программирования. Такой язык называется *псевдокодом.* Запись алгоритма на псевдокоде называется *структурным планом.* Псевдокод удобен тем, что позволяет программисту сосредоточиться на формулировке алгоритма, не задумываясь над синтаксическими особенностями конкретного языка программирования.

При разработке структуры программы удобнее пользоваться записью алгоритма в виде *блок-схемы* (в англоязычіюй литературе используется терми *н flowchart).* Для изображения основных алгоритмических структур и блоков на блок-схемах используют специальные графические символы

Ввод данных

Блок вычислений

Вывод данных

Ветвление

26. Структура программ

Паскаль-программа является текстовым файлом с собственным именем и с расширением .pas. Схематически программа представляется в виде последовательности восьми разделов:

1. Заголовок программы.
2. Описание внешних модулей, процедур и функций.
3. Описание меток.
4. Описание констант.
5. Описание типов переменных.
6. Описание переменных.
7. Описание функций и процедур.
8. Раздел операторов.

11срвым в программе идет зарезервированное слово program. За ним после одного пли нескольких пробелов, следует *идентификатор* — имя программы. Идентифи-

0 а горы могут содержать любое количество символов, но Турбо Паскаль распознает только первые 63 из них, что, разумеется, намного превосходит реальные потребности. Идентификатор должен начинаться с буквы или с символа подчеркивания.

■ (.н ем могут идти буквы, цифры и символы подчеркивания. Взятая в целом, фраза

program stroganova_kate;

ч пляется заголовком программы с именем stroganova_kate. Каждое описание должно миершаться точкой с запятой. Таким образом, первая строка любой программы имеет вид:

program имя_программы;

I! Турбо Паскале оператор заголовка программы может быть опущен. Имя про-

1 1 'л м мы фактически никогда не используется в самой программе, и оно совершенно не связано с именем внешнего файла, содержащего текст программы. Тем не мщее имя программы не должно совпадать с именами переменных

или других

■ *л тентов программы.*

11осле заголовка программы обычно идут описания переменных, констант, меток, подпрограмм и других объектов, используемых в программе. Эта часть программы пллывается *разделом описаний.* В простейших программах эта часть может быть нус гой и не содержать никаких описаний.

1 . 1 ждая программа обязательно должна иметь часть, которая выполняет какие-либо **и in**твия и называется *разделом операторов* (иногда **—** *телом программы).*

Минимально допустимой выполняемой частью программы является *составной оператор'.*

begin

SI; S2: ... ; Sn; end

I лс M,..., Sn — операторы, а зарезервированные слова begin и end играют роль

ско-
..., но только для операторов, а не для математических выражений. Они так и na-

м.шются — **операторными скобками.** *Каждому begin в программе должен соответ- I I пинать end. Обратное, вообще говоря, неверно, так как end может Заканчивать 1*>.| |*дслы, начинающиеся не только с begin, но и с других зарезервированных слов, например case и record. За телом программы должна следовать точка — признак нно, что здесь находится конец программы.*

program ruinm; Раздел описаний
begin
Раздел операторов

11 риведу еще два примера. Операторы в программе могут размещаться как на отдельных строках, так и по несколько в строке. Операторы разделяются точкой с запятой. В первом из приведенных ниже примеров между каждыми двумя последовательными разделителями находится *пустой оператор.* Обратите внимание на то, что допускаются дополнительные пары begi n...end и дополнительные точки с запятой.

program DE; begin
end.
program FG; begin begin
end;
end.

Сделаю замечание по поводу оформления записи текста программы, так как оно важно не только с точки зрения эстетики, но и с точки зрения эффективности труда программиста. При наборе текста программы ее операторы следует располагать таким образом, чтобы была понятной логика работы программы. Для этого используются пробелы между операторами или элементами операторов, а также отступы. Каждый программист имеет свой стиль использования пробелов и отступов. Они помогают читателю программы определить уровень подчиненности каждой строки. Компьютерные программы содержат большое количество информации в концентрированном виде, и хорошее форматирование может оказаться очень полезным. Надо иметь в виду, что вряд ли вновь написанная программа сразу будет работать правильно.

. Комментарий представляет собой текст, который находится между фигурными скобками или между парами символов, состоящими из круглой скобки и звездочки (см. табл. 2.2). I гм г комментария не обрабатывается компилятором и не включается в исполни ' мый файл. Комментарии позволяют включить подробное описание программы и пояснения к ней прямо в исходный текст. Грамотное и уместное применение ком **MI м** гариев упрощает понимание программы, облегчает жизнь ее автору и программистам, работающим с уже готовым текстом. У фигурных скобок есть и нестап- ыртное применение: во время отладки часто возникает необходимость временно V фить из программы какие-то операторы, сохранив, тем не менее, их запись. Про ■ нч'ипий способ — заключить

соответствующий фрагмент программы в фигурные скобки, то есть превратить его в комментарий.

27. Выражение, переменные

Нее переменные, используемые в программе, должны быть перечислены в *разделе описания переменных.* Этот раздел состоит из *предложений описания переменных.*

I аких предложений может быть несколько, размещаются они между заголовком программы, подпрограммы или модуля и зарезервированным словом **begin,** откры- иающим раздел операторов программы, подпрограммы или модуля. Предложения ' шисания переменных могут располагаться вместе (и это одна из составных частей хорошего стиля программирования), но могут и чередоваться с описаниями других объектов: констант, процедур, функций и т. д.

И предложениях описания могут использоваться объекты, описания которых содержатся в других предложениях описания.

ВНИМАНИЕ ---------- ---------- -

Следует соблюдать правило: если в описании используется какой-либо объект, определенный пользователем, описание этого объекта должно предшествовать данному описанию.

Предложение описания переменных имеет вид:

var vl. v2 _ : typejd;

Здесь v 1, v2, список переменных, и котором имени переменных разделяются

нанятыми, a type Id надает ***тип*** переменных на данною списка и является идентификатором типа. Если н программе используются переменные ранных типов, то в предложении var приводятся списки имен переменных каждого типа:

var v_l_l, v_l_2 _ :type_1d_l:
v_2_l. v_2_2 _ :type_id_2;
v_n_l, v_n_2 _ :type_1d_n:

Пример описания переменных: **var**
cows, sheeps : Word; overman : Real:
milkmaid : Extended:

Здесь Word, Real и Extended — названия типов.

28.Ключевые слова языка

В дальнейшем мы разберем применение приведенных в данной таблице зарезервированных слов, а сейчас лишь кратко поясним смысл некоторых из

них, наиболее часто используемых в программах на Паскале.

- Заголовки, то есть первые операторы программ и библиотечных модулей, — это program и unit.

Для описания переменных, констант и составных частей программы — подпрограмм-процедур и подпрограмм-функций — используются зарезервированные слова var, const, procedure и function.

- Операторы описания типов переменных, задаваемых пользователем, — это type, array, string, record...end, file of.

Слова, используемые для программирования составных операторов, а также начинающие и оканчивающие последовательность исполняемых операторов программы, begin и end.

Алфавит, специальные символы и специальные слова языка Паскаль

- Операторами, управляющими ходом выполнения программы (они так и на зываются — *управляющие операторы),* являются if...then...else, for...to...do, repeat...unti 1, case...of...end, for...downto...do, whi 1 e...do.
- В библиотечных модулях используются зарезервированные слова i mpl ementati on, interface.

О Зарезервированные слова для обозначения арифметических и логических оно раций — di v, mod, shl, shr, and, or, not и некоторые другие.

U В программах, написанных с использованием методов объектно-ориентирова]ню го программирования, применяются зарезервированные слова object, constructor, destructor, public и virtual.

absolute	end	near	shr
and	external	nil	string
array	far	not	then
assembler	file	object	to
begin	for	of	type
break	function	or	unit
case	goto	packed	until
const	if	private	uses
constructor	implementation	procedure	var
continue	in	program	virtual
dostructor	inline	public	while
div	interface	record	with
do	interrupt	repeat	xor
down to	label	set	
also	mod	shl	

Кроме перечисленных зарезервированных слов есть ключевые слова, которые нс пользуются для обозначения типов переменных или других объектов програм мы. Забегая вперед, приведу пример программы показывающий, что в определенных ситуациях допускается использование ключевых слов в ка честве имен переменных (переменные **Real** и **Integer).** Использовать так зарезер иированные слова категорически запрещено! Да и применение ключевых слои по по назначению считается очень плохим стилем программирования. Его еде дует избегать

29.Простые операторы; присвоение,обращения к процедурам

I. <10 каждой программы или подпрограммы состоит из последовательности оно **риторов,** каждый из которых выполняет определенное действие. Рассмотрим по I вторые операторы языка Паскаль. Начнем с *оператора присваивания.*

variable expression;

Иырлжение справа от символа присваивания состоит из констант, переменных,

| вращений к функциям и знаков операций. Вначале вычисляется значение выражения («правое» значение переменной). Затем полученное значение заносится и ячейку памяти компьютера (ее адрес является «левым» значением переменной), м резервированную под переменную, имя которой указано в левой части опера ю- ,присваивания. Значение выражения должно быть совместимо по типу с указам i,iHi переменной. Например, любые значения целого типа могут быть присвоены другим переменным целого типа; значение любого выражения целого типа можем 111.n il присвоено любой переменной вещественного типа и т. д.

Примеры операторов присваивания:

a1 := 0.5; y X / (1.0 + x);

Процедуры можно считать нестандартными операторами языка. Процедура программируется пользователем или содержится в готовом наборе процедур *{библиотеке, библиотечном модуле)* и обычно выполняет достаточно сложное действие. Процедура вызывается путем указания ее имени, если она не имеет параметров, или ее имени с параметрами, заключенными в круглые скобки. Тем самым инициируются действия, объем которых может быть довольно большим. Примерами вызовов процедур являются:

WriteLn:

Readdnfile. x);

Программирование процедур обсуждается в главе 3.

30.Структурные и сложные операторы

Реализация последовательности действий (структуры следования) выполняется с помощью *составного оператора*:

begin **Споследовательность операторов>** end;

Составной оператор - это оператор вида:

```
begin
```

S_l:

S_2:
S_n;
end:
где операторы S_l, S_2 S_n, в свою очередь, могут быть простыми или составны
ми операторами. Составной оператор трактуется как один оператор. Такая конструкция необходима в ситуациях, когда, согласно формальным правилам языка, разрешается использовать лишь один оператор, а в действительности их требуется несколько. С этим приходится сталкиваться, например, при программировании циклов или условных операторов.
Наконец, несколько слов о знаках пунктуации. Строго говоря, символ «точка с запятой» в Паскале является не *ограничителем*, *& разделителем* операторов. Поэтому точка с запятой после S_n необязательна. Тем не менее, использование точки с запятой облегчает модификацию программы, например при включении добавочных операторов.
Пример составного оператора: begin
S := S + 1: a := S - Sqr(S);
end;
Структурные операторы составляются из специальных зарезервированных слов, логических выражений и других операторов. Каждый такой оператор явно или неявно содержит одну или несколько логических проверок.
1.Оператор if...then
Оператор i f...then называется *условным оператором* и имеет вид: if expressionl then statementl:
2.Оператор if...then...else
Этот оператор является полной версией условного оператора и имеет вид: if expression then statementl else statement2;
Вложенные операторы if...then...else
Как уже отмечалось, условные операторы можно «вкладывать» друг в друга, программируя таким образом сложные (*многовариантные*) ветвления
3. Оператор case...of...end
и „я ситуаций, где имеется несколько (три и более) альтернатив, больше подходит ратор case. Этот оператор реализует многовариантное ветвление и называется *итератором выбора*

31.Операторы условия

Для реализации развилки в Паскале предусмотрены два оператора: *условный оператор и оператор варианта* (выбора). Первый выглядит так:
if Слогическое выражение> then <оператор1> else <оператор2>; или
if Слогическое выражение> then <оператор>;.
Оператор варианта имеет следующую форму:
case <выражение> of
Ссписок констант 1>: Соператор 1>;

Ссписок констант 2>: Соператор 2>;
Ссписок констант N>: Соператор N>
end; .

Оператор if...then

Оператор i f...then называется *условным оператором* и имеет вид: if expressionl then statementl:

Здесь выражение expressionl является *логическим.* Логическое выражение принимает одно из двух возможных значений — True (истина) или False (ложь). Часто в роли логического выражения выступает какое-то условие, которое может выполняться либо не выполняться. В первом случае его значение — «истина», а во втором — «ложь». Программирование логических выражений мы будем разбирать позже. Если логическое выражение expressionl принимает значение «истина», то выполняется оператор statementl. В противном случае выполняться будет оператор, следующий за данным логическим оператором.

Следует отметить, что, согласно формальным правилам языка, в условном операторе после then допускается применение *только одного* оператора. Но в практике программирования чаще возникают ситуации, когда при выполнении условия в логическом выражении expressionl следует выполнить *несколько* операторов языка. Решается эта проблема, как уже было сказано, применением составного оператора.

Операторы if...then можно «вкладывать» друг в друга, так как конструкция
if expression2 then statement2:
также является оператором и может заместить оператор statementl:
1f expressionl then If express1on2 then statement2;
Исполняемые операторы
Пример условного оператора:
if Centigrade = 0 then Write('Температура замерзания воды'):

Условный оператор реализует алгоритмическую конструкцию «ветвление», точнее
говоря, *бинарное ветвление*, когда выполнение алгоритма в зависимо сти от выполнения или невыполнения условия может пойти по одной из двух ветвей.

Оператор if...then...else

Этот оператор является полной версией условного оператора и имеет вид: if expression then statementl else statement2;

Выполняется данный оператор следующим образом: если выражение expression принимает значение «истина», то управление передается на оператор statementl, если же нет, то на оператор statement2.

11 риведу ошибочный вариант данного оператора: if expression then statementl else; statement2; i 1десь первая точка с запятой завершает оператор i f... then., .el se, не выполняя никаких действий в случае el se, а затем (в любом случае) выполняется оператор statement2. (Синтаксис этого примера формально правильный, но при этом нарушав! ся лш пкд выполнения алгоритма и появляется *логическая ошибка.* Искать логические ошиб ки в программе

гораздо труднее, чем синтаксические. Ведь синтаксис проверяется .ттоматически транслятором, а с логикой работы программы придется разбирд і ь си самостоятельно.

Следующий оператор допускает двоякую интерпретацию:

```
1f expressionl then if expression2 then statement2
else
statementl:
```

І Ь'рвый вариант соответствует такой последовательности операторов:

```
1f expressionl then begin
if expression2 then statement2
   else
   statementl; end;
```

111 (>рой вариант выглядит так:

```
If expressionl then begin
   1f expression2 then statement2 end else
   statementl:
```

Компилятор Паскаля всегда выбирает первый из приведенных вариантов — каж лому else соответствует *ближайший* предшествующий if. Если требуется реализа

ия второго варианта, можно использовать операторные скобки begin...end. В об- 1см случае, чтобы четко определить, что чему подчинено, используйте begin...end »алогично круглым скобкам в арифметических выражениях. Можно также ис- ользовать следующую конструкцию:

```
if expressionl then if expression2 then statement2
else
else
statement].:
```

) ra конструкция, правда, менее наглядна, чем вариант с операторными скобками.

ı ример условного оператора:

```
1f Two = 2 then
WriteLn('Два равно 2')
else
WriteLnf'Это не 2. В чей дело?'):
```

Вложенные операторы if...then...else

Как уже отмечалось, условные операторы можно «вкладывать» друг в друга, программируя таким образом сложные (*многовариантные*) ветвления. Рассмотрим следующий оператор:

```
if expression^ then statement^
else
1f expression_2 then statement_2
else if expression_3 then statement_3
else if expression^ then statement ji :
```

Здесь вначале вычисляется значение логического выражения expression_l. Если оно истинно, выполняется оператор statement_l, если же это значение

ложно, вычисляется значение выражения expression_2. В том случае, когда полученное значение истинно, будет выполняться оператор statement^, при значении «ложь» будет вычисляться выражение expression_3, и т. д.

Если выражения expressionj независимы, то есть вычисление их значений в любом порядке дает один и тот же результат для каждого из них, имеет смысл располагать их в таком порядке, чтобы выражение, с наибольшей вероятностью принимающее значение «истина», стояло на первом месте, выражение, принимающее значение «истина» с меньшей вероятностью, - на втором и т. д. Это уменьшит время выполнения данного фрагмента программы, особенно если вложенный оператор появляется в цикле, который выполняется многократно.

II ример вложенных условных операторов:

```
1f Two - 2 then if One - 1 then
WriteLnf'Единица раина Г)
```

I ti i юлняемые операторы

```
else
WriteLnCЕдиница не равна Г)
else
if Three = 3 then
WriteLnC'Три равно 3')
else
WriteLnC'Три не равно 3');
```

Оператор case...of...end

и „я ситуаций, где имеется несколько (три и более) альтернатив, больше подходит ратор case. Этот оператор реализует многовариантное ветвление и называется *итератором выбора.* Он имеет следующий вид.

```
case expression of
va1ues_l : statement^; values_2 : statement_2:
    values_n : statement jr. else statement: end:
```

Рассмотрим элементы этой конструкции. Во-первых, в нее входят три зарезерви- I,.пенных слова: case, of и end. Между case и of находится выражение expression,

„трое может принимать (но не обязательно) одно из значении, перечисленных и списках значений valuesj, valuesj, ..., values_n, 11:1Х0ДЯЩИХС^;^^Г“^ •mi, Данное выражение называется *селектором* оператора case. Каждый оператор, „пущий за двоеточием, отделяется от следующего списка значении точкой с запя- mii 11ри выполнении данного оператора вначале вычисляется значение селект , м ■ (аТем выбирается тот список значений, которому принадлежит полученное чгпие^ивыполняется соответствующий оператор. Ветвь el se, которм соот^тстает „к/юму значению выражения expression, не включенному в списки, может быт

опущена.

11 списках значений оператора case допустимыми являются типы переменных, на- н.птемые *скалярными* (они будут обсуждаться позже),

„сщгтвенные типы. Любое заданное значение селектора может входить спис

Лий неоднократно, но выполняться будет лишь первая , ,м(ч,у что «стилистически» подобная конструкция выглядит не очень изящшх I , значение селектора отсутствует в списках значении, ни одна из альтерната ,,толпиться не будет. В этом случае выполняется ветвь else или, если эта ветвь отсутствует, следующий за case оператор.

I („пеню применение данного оператора следующим примером. Пусть необходимо преобразовать целое число *N* в зависимости от величины остатка от его делени п.I 17 следующим образом: il если *N mod* 17 = 0, то *N*= 0;

j если *N* mod 17 = 1 или 6, то *N*= *-N;*

I если *N* mod 17 = 2,3 или 5, то *N* = 2 x JV;

если *N* mod 17 " 4, то *N"* 3 x *N*

1 1 во всех прочих случаях *N" 5* x *N.*

Решение этой задачи на Паскале выглядит следующим образом:

```
case N mod 17 of

0       N:=0
1. 6    N ;=  1;
                *
2. 3.5  N:=2 N;
                *
4       N:=3 N:
else    N:=5 N;

end:
```

В данном примере селектором является выражение N mod 17. Кроме того, имеются четыре списка значений и ветвь el se.

32.Операторы цикла

Для реализации *циклов* в Паскале имеются три оператора.

Цикл с предусловием:

while Слогическое выражение> do Соператор>;.

Действие: вычисляется значение логического выражения. Если оно равно *true*, то выполняется оператор, после чего снова вычисляется значение логического выражения, в противном случае действие заканчивается.

Цикл с постусловием:

repeat

Споследовательность операторов>

until Слогическое выражение>;

Действие: выполняется последовательность операторов. Далее вычисляется значение логического выражения. Если оно равно *true,* то действие заканчивается, иначе снова выполняется последовательность операторов и т.д.

Цикл с параметром:
for Спараметр>:= Свыражение 1> to Свыражение 2> do Соператор>
Параметр, выражение 1, выражение 2 должны быть одного ординального типа. Параметр в этом цикле возрастает.

for Спараметр>:^Свыражение 1> downto Свыражение 2> do Соператор>.

Параметр в этом цикле убывает.

Оператор цикла с предусловием: while...do

Оператор цикла является важнейшим оператором и имеется в большинстве современных языков программирования, а сама идея цикла возникла еще в XIX веке. Цикл позволяет многократно выполнить некоторую последовательность действий, которая задается операторами, составляющими *тело цикла.* В Паскале имеется несколько разновидностей оператора цикла.

ПРИМЕЧАНИЕ---------- ---------- —--------

Термин «цикл» был введен леди Адой Лавлейс (1815-1852), дочерью поэта Джорджа Байрона, которая считается первым программистом, точнее программисткой. Будучи соратником и спонсором Бэббиджа, она занималась разработкой программ для его машин и предложила ряд терминов и идей, которые используются в программировании и в **настоящее время. Среди этих идей следует упомянуть применение перфокарт, условные переходы, циклы, ячейки памяти. Бэббидж и Лавлейс подошли к таким важным понятиям, как подпрограмма, библиотека подпрограмм и некоторые другие. Термин «библиотека» был введен Бэббиджем, а термины «рабочая ячейка» и «цикл» — Адой Лавлейс.**

Начнем с *оператора цикла с предусловием.* Данный оператор имеет вид:

while expression do statement;

При выполнении этого оператора вначале вычисляется значение логического выражения expression. Если это значение истинно, выполняется оператор statement. Затем значение выражения проверяется вновь, и все повторяется до тех пор, пока выражение не примет значение «ложь». Каждое выполнение цикла иногда называют *итерацией цикла.* Если выражение принимает значение «ложь» при первой же проверке, то оператор statement не выполняется вообще.

Особо отмечу частный случай: while True do statement;

Здесь оператор statement будет выполняться бесконечно. Если цикл выполняется бесконечно, говорят о «зацикливании» программы. Прервать работу зациклившейся программы можно средствами операционной системы.

Пример оператора цикла с предусловием:

```
Counter 0;
while Counter < 10 do begin
Wr1 tot'Значение счетчика равно Counter);
```

Игпопняемые операторы

```
Counter + 2;
WriteLn:
```

Counter

ся их инициализация.

Оператор цикла с постусловием: repeat...until

Оператор ***цикла с постусловием*** имеет вид. repeat statement until expession:

$^{K}0$~— мож$_{ет}$ быть любы», в то» числе „ составным оператором:

begin

statements:

statement_2: statement^:

I, Гкле repeat .until операторные «юбки be$_{9}$1n...end могут быть опущены. Таки» !/;Г». в Общем случае оператор repeat...untl 1 имеет следующий вид.

repeat

statement^:

statement_2:

statements:

Точка с запятой перед зарезервированным словом until необязательна. В приведенном ниже частном случае цикл выполняется бесконечно.

repeat

statements: statements:

statements:

и,1 1 **пример** цикла с постусловием:

Count 0:

repeat , r ...

Write('Значение счетчика равно . иоиптл.

WriteLn;

Count Count + 2; until Count - 10;

Операторы цикла со счетчиком for...to...do и for...downto...do

I ротий вариант оператора цикла — *цикл со счетчиком,* или цикл с параметром. Можно считать, что есть две очень похожих друг на друга разновидности цикла со счетчиком. Первый из этих операторов имеет вид: for j expressionl to expression do statement:

. Здесь переменная j, называемая *управляющей переменной* цикла for, является произвольным идентификатором, который объявляется как переменная любого скалярного типа. К скалярным относятся целый, символьный, булев и перечислимые типы.

При выполнении оператора for сначала вычисляется значение выражения expressionl, затем вычисляется значение выражения expressi оп2, далее управляющая переменная цикла последовательно пробегает все значения от expressionl до ex- pression2. В том случае, если значение expressionl сразу оказывается больше зна- ^{,1ИЯ} expression2, тело цикла не будет выполняться вовсе. Значения expressionl п expression2 остаются неизменными в ходе выполнения всего цикла for.

Рассматриваемый вариант цикла for эквивалентен следующей последовательно- с I и операторов (в предположении, что при каждом выполнении оператора statement не изменяются значения j и к):

j expressionl: к :- exprèssion2: while j <= к do begin
statement:
Inc(j):
end:
Оператор for вида

for j := expressionl to expression2 do statement: не эквивалентен последовательности операторов begin
j := expressionl:
while j <= expression2 do
begin

statement : j := j + 1: end; end:

потому что выражение expression2 может изменяться при каждом выполнении оператора statement в цикле while.

В теле цикла for следует избегать операторов, изменяющих значение управляющей переменной j. Несмотря на то что использование подобных конструкций не приводит к ошибкам компиляции, они потенциально опасны и могут приводить к неприятным последствиям. Рассмотрим пример:

sum := 0:
for k := 1 to 100 do begin
turn :• sum + Sqr(k); к к + 2:
m id,

• I иг фрагмент программы является попыткой просуммировать $п^2$ повеем целым ж I ч « **'1111** я м вида *n* = 3/e +1, лежащим в диапазоне от 1 до 100. Синтаксис данного примера формально правильный. Здесь допущена ошибка реализации алгоритма,

ı и I а к управляющая переменная к изменяется в составном операторе, управляемым гой же переменной к. Правильной будет конструкция следующего вида:

мин : — 0 :
for к 0 to 33 do sum := sum + Sqr(3 * к + 1):
ИЛИ
мин 0: к := 1: repeat

sum := sum + Sqr(k): к к + 3: until к > 100:

Uni пс выполнения цикла for значение управляющей переменной становится не- нмределенным.

Пирпант for...downto...do... цикла for аналогичен циклу for...to...do, за исключением ино, что в нем управляющая переменная на каждом шаге выполнения не увеличи- N и гея, а *уменьшается* на единицу:

fur j := expressionl downto expression do statement;

11**11**ДНОДЯ итоги, для применения циклов можно сформулировать следующие рекомендации.

| 11с пользуйте цикл for в том случае, когда точно знаете, сколько раз должно быть выполнено тело цикла. В противном случае обратитесь к циклам repeat или while.

ıı Используйте repeat, если необходимо, чтобы тело цикла выполнялось по

крайней мере, один раз, а число повторений цикла заранее не известно.
11 Используйте whi 1 e, если хотите, чтобы проверка была произведена прежде, чем будет выполняться тело цикла, а число повторений цикла заранее не известно. Иногда бывает удобно проводить проверку на возможный выход из цикла где- нибудь в его середине, а не в начале или в конце. Такой выход из цикла обеспечи- **11**. и "гея вызовом процедуры Break модуля System, которая прерывает выполнение I .iMoro внутреннего вложенного цикла, будь то for, while или repeat. Указанный модуль подключается к программе автоматически, если в этом есть необходимость. Пример:

```
while True do begin
    statementl;
    if expression then Break; statement2; end:
```

(Следует также упомянуть процедуру Conti nue, которая прерывает выполнение тела самого внутреннего цикла for, while или repeat и передает управление на его заголовок, так что начинается выполнение очередной итерации цикла.

33. Массивы

Массив — это последовательность, состоящая из фиксированного числа однотипных элементов:

type <имя типа> = array ссписок типов индексов> of <тип элементов>.

Число типов индексов называется размерностью массива. После описания типа массива конкретные массивы можно задать в разделе описания переменных, например:

```
type vector = array [1..10] of real; table = array [1A1..1Z1,1..5] of integer;
var a,b: vector; c: table;
```

Описание массива—типа можно совместить с описанием соответствующих переменных:

```
var a,b: array [1..10] of real; d: array [byte] of char; .
```

Строковый тип определяется в разделе описания типов, переменные этого типа - в разделе описания переменных:

```
type word: string[20];
var a,b,c: word;
```

или

```
var a,b,c: string[20];
d: string[30];
```

Для строковых величин определены четыре *стандартные функции:*

1. Функция соединения — **concat (si, s2,** . . ., **sk)**.
2. Функция выделения — **copy (s, i, k)**. Из строки *s* выделяется *к* символов, начиная с /-го символа.
3. Функция определения длины строки — **length (s)**.

4. Функция определения позиции — **pos(s,t).** Вычисляется номер позиции, начиная с которой строка *s* входит первый раз в строку *t*; результат равен 0, если строка *s* не входит в *L*

Также определены четыре *стандартные процедуры* для обработки строковых величин:

1. Процедура удаления — **delete (s, i, k)**. Из строки *s* удаляется *к* символов, начиная с /-го символа.
2. Процедура вставки — **insert (s, t, i)**. Строка *s* вставляется в строку *t*, начиная с позиции /.
3. Процедура преобразования числа в строку символов — **str (k, s)**.
4. Процедура преобразования строки из цифр в число — **val (s, k, i)**.

34. Простые и текстовые типы данных: ввод и вывод данных

К ним относятся: real — вещественный; integer — целый; boolean — логический; byte — байтовый; char — символьный.

Перечисляемые и интервальные типы данных задаются с помощью простых типов. Описание перечисляемого типа выполняется в разделе типов по схеме:

type <имя типа> = ссписок имен> .

Примеры:

type *operator*=(plus,minus,multi,divide);

color=(white,red,blue,yellow,purple,green);

Интервальный тип — это подмножество другого уже определенного простого типа, называемого базовым. Пример:

type days = (mon,tue,wed,thu,fri,sat,sun); workdays = mon..fri;

index = 1. . 30; .

letter = 'a'..'z';

Можно задать интервал и в разделе переменных: var a:1..100; Ь: -25..25/.

35. Процедуры

Подпрограммы являются основными строительными блоками, из которых, как из кирпичиков, собирается любая более или менее серьезная и большая программа. Есть ряд причин, благодаря которым в языках программирования появились и получили распространение подпрограммы. Прежде всего, часто оказывается, что определенная последовательность операций выполняется неоднократно и при атом, возможно, для разных значений входных параметров. Такую последовательность можно считать обобщенной операцией. В этом случае удобно один раз описать последовательность операций, присвоив этому описанию имя, и в дальнейшем использовать это имя в качестве нового оператора. Это существенно экономит время, необходимое для разработки программы, и сокращает размер ее секста. Можно считать, что подпрограммы дают программисту возможность

дополнить язык своими собственными командами. Надежные и эффективные подпрограммы могут использоваться и другими программистами. Использование подпрограмм позволяет повысить уровень абстракции кода программы, так как вместо детального описания последовательности операций применяется ее название.

Другое применение подпрограмм заключается в том, что при разработке большой программы ее можно разбить на отдельные блоки, каждый из которых выполняет логически завершенную часть общей работы. Каждый блок можно оформить в виде одной или нескольких подпрограмм. Если создается большая и сложная программа, ее удобно спроектировать на уровне подпрограмм, и только затем перейти к детальной проработке каждой подпрограммы. Разработку разных подпрограмм можно поручить разным группам программистов. Работая одновременно, они быстрее справятся с задачей в целом, а отладка каждой части окажется более простым делом, ведь ее объем значительно меньше, чем объем целой задачи.

В главе 3 мы познакомились с подпрограммами-функциями, описание которых создается с помощью зарезервированного слова functi on. Подпрограмму-функцию можно определить как фрагмент программы или модуля, который начинается заголовком function и заканчивается зарезервированным словом end. Между ними находятся раздел описаний и раздел операторов. При вызове функция возвращает единственное значение, носителем которого является ее имя (рис. 4.1). Функция может быть только простого или строкового типа или типа «указатель», поэтому возможности функций ограничены. Ведь иногда результатом работы подпрограммы должен быть целый *набор* значений, в том числе и разнотипных, а иногда ее выполнение не сводится только к вычислениям (например, если необходимо вывести результат на устройство печати).

Обмен данными между вызывающей программной единицей и подпрограммой-функцией

Подпрограммы-процедуры могут выполнять все виды действий, включая вычисление одного или более результатов

36. Системное програмное обеспечение.

Программы, относящиеся к данному классу, предназначены для управления работой аппаратной части компьютеров и компьютерных сетей. Они, как правило, не решают конкретных задач пользователя, но создают условия для их решения. С их помощью решаются следующие задачи:

- обеспечение устойчивой работы компьютера и вычислительной сети;
- создание условий для нормальной работы прикладных программ;
- диагностика и профилактика аппаратной части компьютера и вычислительной сети;
- выполнение вспомогательных технологических операций (резервное копирование, архивирование, восстановление файлов).

Все множество системных программ можно разделить на две большие группы: базовое и сервисное системное ПО. Базовое программное обеспечение — это минимальный набор программных средств, обеспечивающих работу компьютера [8]. Эти программы, как правило, поставляются вместе с компьютером. В этот подкласс ПО обычно включают операционные системы (ОС), операционные оболочки, сетевые операционные системы.

Операционная система — это комплекс программ, предназначенных для управления выполнением пользовательских программ, планирования и управления вычислительными ресурсами ПК.

Операционные оболочки — специальные программы, предназначенные для облегчения общения пользователя с командами операционной системы. Операционные оболочки имеют текстовый и графический варианты интерфейса конечного пользователя.

Сетевые операционные системы — комплекс программ, обеспечивающих обработку, передачу и хранение данных в сети.

К подклассу сервисного программного обеспечения относятся

программы, предназначенные для обслуживания компьютера. Эти программы расширяют возможности базового ПО. Иногда некоторые сервисные программы включают в состав операционной системы, но, как правило, они устанавливаются дополнительно. По функциональному признаку среди сервисного ПО можно выделить:

- программы диагностики работоспособности компьютера;
- программы обслуживания дисков, обеспечивающие проверку качества поверхности магнитного диска, контроль сохранности файловой системы, сжатие дисков, создание страховых копий дисков, резервирование данных на внешних носителях и др.;

- антивирусные программы, обеспечивающие защиту компьютера, обнаружение и восстановление зараженных файлов;
- программы архивирования данных, которые обеспечивают процесс сжатия информации в файлах с целью уменьшения объема памяти для ее хранения;
- программы обслуживания сети.

Программы, служащие для выполнения вспомогательных операций обработки данных или обслуживания компьютеров (диагностики, тестирования аппаратных и программных средств, оптимизации использования дискового пространства, восстановления разрушенной на магнитном диске информации и т. п.), называют утилитами.

37. Прикладное програмное обеспечение.

Прикладное ПО предназначено для решения прикладных задач, т. е. непосредственно обеспечивает выполнение необходимых пользователю задач на компьютере. Условно среди множества прикладных программ по их назначению можно выделить следующие подклассы:

- программы обработки текстов;
- графические редакторы;
- программы обработки фото- и видеоизображений;
- системы деловой и презентационной графики, позволяющие наглядно представить на экране различные числовые данные и зависимости между ними;
- системы научной и инженерной графики, позволяющие автоматизировать весь комплекс работ по созданию чертежей, схем, графиков, создавать объекты трехмерной графики и т. п.;
- электронные таблицы;
- системы управления базами данных;
- информационно-поисковые системы;
- сетевое программное обеспечение. Оно позволяет не только объединить компьютеры в сети различного ранга, но и облегчает навигацию по таким сетям. К нему же относятся и программы работы с электронной почтой, доступа к видеоконференциям, браузеры Интернет и т. п.;
- игровые программы.

Очень часто программы различных типов объединяются в интегрированные пакеты, что чрезвычайно упрощает обмен информацией между ними. Так, пакет Microsoft Office состоит из текстового редактора, простого графического редактора, электронной таблицы, системы разработки и управления базами данных, программы электронной почты, программы разработки Интернет-страниц, программы просмотра Интернет-страниц, программы создания презентационной графики.

Кроме того, даже отдельные программы часто включают в себя элементы программ другого типа. Так, почти все игровые программы снабжены сетевыми компонентами, позволяющими подключить к игре несколько

компьютеров.

38. Классификация ОС.

Для классификации операционных систем может быть использовано несколько признаков. В настоящее время чаще всего используются следующие:

1) количество одновременно работающих пользователей;
2) число задач, одновременно выполняемых под управлением ОС;
3) тип пользовательского интерфейса;
4) способ использования аппаратных и программных ресурсов;
5) количество используемых в компьютере процессоров;
6) разрядность процессора.

Рассмотрим некоторые из особенностей ОС в зависимости от того или иного классификационного признака. В зависимости от количества пользователей, которые могут для решения своих задач одновременно обратиться к одному и тому же компьютеру, различают однопользовательские и многопользовательские ОС. Главным отличием многопользовательских ОС от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей. Каждому пользователю выделяется свой сегмент оперативной памяти.

По числу задач, одновременно выполняемых под управлением ОС, они делятся на однозадачные и многозадачные. В многозадачном режиме каждой задаче (программе, приложению) по-
очередно выделяется какая-то доля процессорного времени. По Поскольку процесс переключения идет очень быстро, а выделяемые задачам доли процессорного времени достаточно малы, то у пользователя создается впечатление одновременного выполнения сразу нескольких задач. При использовании многозадачной ОС можно одновременно организовать вычисления в среде табличного процессора, включить принтер для печати текста, запустить проигрыватель музыкальных произведений, вести поиск вирусов и рисовать в графическом редакторе.

Современные операционные системы обеспечивают реализацию двух типов пользовательского интерфейса — текстового и графического. Текстовые (неграфические) операционные системы предоставляют пользователю интерфейс командной строки. Основным устройством управления в данном случае является клавиатура. Управляющие команды вводят в поле командной строки, где их можно и редактировать. Исполнение команды начинается после ее утверждения, например нажатием клавиши **ENTER.**

Графические операционные системы дают возможность использовать

более сложный тип интерфейса, в котором в качестве органа управления, кроме клавиатуры, может использоваться устройство позиционирования (мышь, трекбол, сенсорная панель). Работа с графической операционной системой основана на взаимодействии активных и пассивных экранных элементов управления. В качестве активного элемента управления выступает указатель манипулятора — графический объект, перемещение которого на экране синхронизировано с перемещением органа управления. В качестве пассивных элементов управления выступают графические объекты на экране монитора (экранные кнопки, значки, переключатели, флажки, раскрывающиеся списки, строки меню и многие другие). Характер взаимодействия между активными и пассивными элементами управления выбирает сам пользователь.

По способу использования аппаратных и программных ресурсов различаются сетевые и локальные операционные системы. Сетевые операционные системы предназначены для эффективного решения задач распределенной обработки данных. Такая обработка ведется не на отдельном компьютере, а на нескольких компьютерах, объединенных сетью. Сетевые операционные системы поддерживают распределенное выполнение процессов, их взаимодействие, обмен данными между ЭВМ, доступ пользователей к общим ресурсам и другие функции, которые превращают распределенную в пространстве систему в целостную многопользовательскую систему. Все сетевые операционные системы делятся на две группы: одноранговые ОС и ОС с выделенными серверами. В одноранговых сетях каждая ЭВМ может выполнять как функции сервера, так и рабочей станции, а в сетях с выделенными серверами реализуется разделение функций: рабочие станции не предоставляют свои ресурсы для других ЭВМ. Услуги предоставляют только серверы.

Что касается других классификационных признаков, то здесь можно ограничиться только констатацией факта, что по количеству используемых в компьютере процессоров ОС могут быть одно- и многопроцессорными; а по разрядности процессора — 8-, 16-, 32- и 64-разрядными.

Очевидно, что процесс совершенствования аппаратных средств сопровождался эволюцией операционных систем. Из всего множества наибольшую известность получили следующие ОС: CP/M, MS-DOS, OS/2, Windows, UNIX и MacOS (для компьютеров Macintosh фирмы Apple).

ОС CP/M была создана для работы на 8-разрядных ПЭВМ. Первоначальный успех этой операционной системы в значительной степени был обусловлен ее предельной простотой и компактностью. Первая версия занимала всего 4 Кбайт. Компактность была весьма важна в условиях ограниченных объемов памяти первых персональных ЭВМ (ПЭВМ).

Операционная система MS-DOS (фирма Microsoft) появилась в 1981 г. В настоящее время существуют версии 6.22 и 7.0 (в составе Windows 95), а также ее разновидности других фирм-разработчиков (DR DOS, PC DOS). Начиная с 1996 г. MS-DOS распространяется в виде Windows 95 — 32-разрядной многозадачной и многопоточной операционной системы с гра-

фическим интерфейсом и расширенными сетевыми возможностями. Операционная система MS-DOS является промышленным стандартом для 16-разрядных ЭВМ на основе микропроцессоров 8086...80486. Все программы MS-DOS хранятся на магнитных дисках, поэтому она называется дисковой операционной системой (Microsoft Disk Operating System).

Операционная система OS/2 разработана фирмой IBM для персональных компьютеров на основе системной прикладной архитектуры, ранее используемой для больших ЭВМ. Это многозаданная, однопользовательская, высоконадежная операционная система, обеспечивающая как текстовый, так и графический интерфейс пользователя, одновременную обработку нескольких приложений, 32-разрядную обработку данных. Важной особенностью операционной системы OS/2 является высокопроизводительная тельная файловая система HPFS (High Performance File System),

имеющая преимущества для серверов баз данных (в отличие от MS-DOS поддерживаются длинные имена файлов), поддержка мультипроцессорной обработки — до 16 процессоров типа INTEL и PowerPC. Версия OS/2 Warp работает с мультисредой и имеет встроенный доступ в сеть Интернет, систему распознавания речи Voice Type, интегрированную версию Lotus Notes Mail для передачи почты через Интернет.

Перспективной является многопользовательская и многозадачная операционная система Unix, созданная корпорацией Bell Laboratory. Данная операционная система реализует принцип открытых систем и широкие возможности по комплексированию в составе одной вычислительной системы разнородных технических и программных средств. Unix обладает наиболее важными качествами, такими, как переносимость прикладных программ с одного компьютера на другой и поддержка распределенной обработки данных в сети ЭВМ. Unix получила распространение для суперкомпьютеров, рабочих станций и профессиональных персональных компьютеров, имеет большое количество версий, разработанных различными фирмами. Согласно прогнозам объем мирового рынка вычислительных систем, базирующихся на ОС Unix, существенно будет возрастать, особенно с переходом к сетевым технологиям.

Операционная система MacOS была разработана для компьютеров Macintosh фирмы Apple. В настоящее время эти компьютеры с MacOS в России используются в основном для так называемой допечатной подготовки полиграфической продукции — книг, журналов, газет. Программы с мощными средствами обработки графики Adobe Photoshop, Adobe Illustrator, Adobe Pagemaker в свое время разрабатывались для этой ОС и только впоследствии были адаптированы под Windows.

Среди сетевых операционных систем в течение длительного времени наиболее широко используемыми были различные версии операционной системы Netware, разработанные фирмой Novell. К одноранговым операционным системам можно отне

сти NetWare Lite и Personal NetWare, а к ОС с выделенным сервером — NetWare 2.2, NetWare 3.12, NetWare 4.0 и NetWare 5.0.

Фирма Microsoft выпустила несколько версий сетевых операционных систем: Windows NT 3.51 и 4.0, Windows 2000/ME/ CE/XP.

Центральное место среди сетевых операционных систем занимает UNIX. Большая популярность пришла к UNIX в 1983 г., когда появилась версия 4.2BSD, имевшая сетевые средства TCP/IP, что позволяет использовать эту систему для работы в глобальной сети ARPANET. Классическая ОС UNIX дала жизнь многочисленным своим потомкам, число которых превышает несколько десятков (AIX, SCO, HP-UX, IRIX, Solaris, Linux и др.).

Особого упоминания заслуживает ОС Linux — свободно распространяемая версия операционной системы UNIX для платформ x86, Motorola 68k, Digital Alpha, Spare, Mips и Motorola PowerPC. В Linux не используются никакие части программного обеспечения, принадлежащие каким-либо коммерческим организациям. По этой причине она получила достаточно широкое распространение.

39. Состав операционных систем

Как правило, операционная система содержит в себе следующие компоненты: ядро, файловую систему, диспетчер задач (или планировщик задач), драйверы устройств и различные сервисные программы, упрощающие обслуживание, наладку и оптимизацию работы внешних устройств или оптимизирующие работу самой ОС.

Ядро операционной системы состоит из базовой системы ввода и вывода (BIOS — basic input/output system), модуля расширения базовой системы ввода-вывода (BIO.COM), модуль обработки прерываний (DOS.COM) и командного процессора (COMMAND.COM).

В архитектуре ПЭВМ **базовую систему ввода-вывода** (BIOS) можно рассматривать, с одной стороны, как составную часть аппаратных средств; с другой стороны, BIOS является одним из программных модулей ДОС. ВЮБ находится не на дисках, как все остальные модули, а в ПЗУ, которое установлено внутри системного блока. ВЮБ обеспечивает:

- автоматическое тестирование основных аппаратных компонентов при включении машины. При этом проверяются большие интегральные схемы (БИС) системного модуля, ОЗУ, клавиатура и ее адаптер, адаптеры накопителей на магнитных дисках, адаптеры параллельного и последовательного интерфейсов. Если обнаруживаются ошибки, ВЮБ выдает на экран соответствующие сообщения (обычно в виде условного кода ошибки) и еще извещает об этом пользователя звуковым сигналом. Дальнейшая работа машины при этом прекращается и пользователю нужно принимать меры к устранению выявленной ошибки;
- вызов блока начальной загрузки ООБ. Загрузка ООБ в память происходит в два этапа; сначала ВЮБ загружает с системного диска в память специальный блок начальной загрузки, а затем уже передает на него управление, и тот, в свою очередь, осуществляет загрузку всех остальных

модулей ООБ. Блок начальной загрузки предназначен для просмотра каталога системного диска и проверки того, что первые два файла являются модулями ООБ. При попытке запуска системы с несистемного диска данный блок выдает сообщение об ошибке;

- обслуживание системных вызовов или прерываний.

Поскольку ВЮБ очень тесно связана с особенностями аппаратуры конкретного компьютера, она является общей и неизменяемой частью всех возможных операционных систем для данной модели ПЭВМ. Поэтому для обеспечения возможности управления с помощью ВЮБ определенным набором аппаратных средств создается дополнительный модуль, придающий гибкость операционной системе.

Модуль расширения ВЮБ — это обычная программа, которая при необходимости может быть заменена другой. Наличие модуля расширения дает возможность включения в состав ОС дополнительных подпрограмм, обслуживающих новые внешние устройства.

Модуль обработки прерываний БОБ образует верхний уровень системы, с которым взаимодействует большинство прикладных программ. По этой причине этот модуль ДОС называют основным. Компонентами данного модуля являются подпрограммы, обеспечивающие работу файловой системы, устройств, обслуживание некоторых специальных ситуаций, связанных с завершением программ, их искусственным прерыванием и обработкой ошибок. Некоторые из этих подпрограмм довольно велики по объему. Многие из функций, реализуемых данным модулем 008, используются не только прикладными программами, но и командами, которые обрабатываются командным процессором.

Командный процессор представляет собой программу, осуществляющую перевод команд с языка программирования на язык машинных кодов. Основная функция командного процессора заключается в приеме, анализе и исполнении команд пользователя, обращенных к 008, и в обработке командных файлов.

Команды пользователя иначе называют командами 008. Они служат основным средством общения пользователя с ОС до тех пор, пока не будет вызвана какая-либо прикладная программа (задача), или «надстройка». Команды 008 позволяют готовить диски для работы, копировать файлы, переименовывать их, удалять из каталогов, сменять текущий каталог и текущий накопитель, изменять режим работы дисплея, выводить содержимое текстовых файлов на экран дисплея, на принтер или в коммуникационный канал.

40. Файлы и файловые системы

Файловая система — совокупность программ, входящих в состав операционной системы, которая управляет размещением и доступом к папкам и файлам на диске. Она обеспечивает подготовку магнитных

носителей к работе (разбиение, форматирование, проверка), размещение на носителях файлов и обеспечение доступа к ним различных программ. Файловая система включает в себя программы доступа к магнитным носителям и специальные таблицы, в которых хранится информация о том, на какие части разбит диск, где на диске располагаются файлы и т. п. Файловая система во многом определяет скорость и эффективность работы магнитных носителей, надежность и секретность хранения информации на них.

Записать информацию на магнитный диск можно только тогда, когда он подготовлен определенным образом — отформатирован. Во время форматирования производится разметка диска, в процессе которой на него записывается служебная информация, которая затем используется для записи и чтения команд и данных, коррекции скорости вращения диска. Запись информации осуществляется по дорожкам, причем каждая дорожка разбивается на секторы размером 512 байт.

В процессе форматирования на диске выделяется системная область, которая состоит из трех частей: загрузочного сектора, таблицы размещения файлов и корневого каталога.

Загрузочный сектор (Boot Record) размещается на каждом диске в логическом секторе с номером 0. Он содержит данные о формате диска, а также блок начальной загрузки DOS. Каждый жесткий диск может быть разбит на несколько логических дисков. На жестком диске имеется область, которая называется главной загрузочной записью MBR (Master Boot Record) или главным загрузочным сектором. В MBR указывается, с какого логического диска должна производиться загрузка операционной системы.

Таблица размещения файлов (File Allocation Table — сокращенно FAT) располагается после загрузочного сектора и содержит описание порядка расположения всех файлов в секторах данного диска, а также информацию о дефектных участках диска. За FAT-таблицей следует ее точная копия, что повышает надежность сохранения этой очень важной таблицы.

Необходимость в таблице расположения файлов вызвана тем, что данные и программы хранятся на носителях информации в виде файлов. У каждого файла есть свой адрес, который записывается двоичным числом. Очевидно, что количество адресов, которые могут быть присвоены файлам, зависит от разрядности этого числа. В ряде операционных систем для записи адреса файла использовались двухбайтные (16-разрядные) двоичные числа. Известно, что с помощью 16 битов можно выразить 2^{16} (65 536) разных значений. В этом случае файлам на жестком диске не может быть предоставлено более чем 65 536 разных адресов, следовательно, на такой диск нельзя записать более 65 536 файлов. Необходимость присвоения каждому файлу собственного уникального адреса привела к появлению нового понятия «минимальный размер адресуемого пространства». Для обозначения этого пространства используется термин **«кластер».** Очевидно, что размер кластера зависит от объема диска и количества адресов, используемых для записи файлов (табл. 4.2, 4.3). Иногда для обозначения

файловой системы используют разрядность адресов FAT. Так, достаточно широко распространены файловые системы FAT-16 и FAT-32.

Размер кластера при использовании FAT-16

Объем диска	Размер кластера
Менее 32 Мбайт	512 байт
32 Мбайт...64 Мбайт	1 Кбайт
64 Мбайт... 128	2 Кбайт
128 Мбайт...256	4 Кбайт
256 Мбайт.,.512	8 Кбайт
512 Мбайт... 1 Гбайт	16 Кбайт
1 Гбайт...2 Гбайт	32 Кбайт

Размер кластера при использовании FAT-32

Объем диска	Размер кластера
513 Мбайт...8 Гбайт	4 Кбайт
[8 Гбайт... 16 Гбайт	8 Кбайт
16 Гбайт...32 Гбайт	16 Кбайт
Более 32 Гбайт	32 Кбайт

Наличие FAT приводит к существованию противоречия: с одной стороны, чем меньше кластеры, тем их больше на диске и тем больше требуется места для хранения самой таблицы; с другой — чем больше кластеры, тем меньше таблица, но зато тем больше «теряется» дискового пространства, потому что размер файла обязан быть кратен размеру кластера. Это не вполне рациональный расход рабочего пространства, поскольку любой файл (даже очень маленький) полностью оккупирует весь кластер, которому соответствует только одна адресная запись в таблице размещения файлов. Даже если файл достаточно велик и располагается в нескольких кластерах, все равно в его конце образуется некий остаток, нерационально расходующий целый кластер.

Для современных жестких дисков потери, связанные с неэффективностью файловой системы, весьма значительны и могут составлять от 25 до 40 % полной емкости диска в зависимости от среднего размера хранящихся файлов. С дисками же размером более 2 Гбайт файловая система FAT-16 вообще работать не может.

Начиная с операционной системы Windows 98 используется более совершенная организация файловой системы — FAT32 с 32-разрядными полями в таблице размещения файлов. Для дисков размером до 8 Гбайт эта система обеспечивает размер кластера 4 Кбайт (8 секторов).

Корневой каталог (Root Directory) всегда находится за копией FAT. В корневом каталоге содержится перечень файлов и директорий, находящихся на диске. Непосредственно за корневым каталогом располагаются данные, организованные в файлы. Файл — это набор взаимосвязанных данных,

воспринимаемых компьютером как единое целое, имеющих общее имя, находящихся на магнитном или оптическом диске, магнитной ленте или на другом носителе информации. Файл обычно отождествляют с участком памяти (ВЗУ, ОЗУ, ПЗУ), где размещены логически связанные данные, имеющие общее имя. Файл записывается на носителе информации в двоичных кодах. Для операционной системы файл представляется как совокупность связанных байтов.

Файлы могут содержать в себе любую информацию. Это могут быть как программы, выполняемые под управлением какой-либо операционной системы, либо файлы с данными для этих программ. Независимо от операционных систем персональных компьютеров все файлы можно разделить на текстовые и бинарные (двоичные) файлы. Текстовыми файлами называют файлы, в которых используются в качестве информационных символы с шестнадцатеричными кодами 20h—7ЕЬ (32—126 десятичными) и 80h—7ЕЬ (128—254 десятичными).

Среди всех текстовых файлов можно выделить подмножество чистых ASCII-файлов, информационные символы которых имеют только коды с номерами 20h—7ЕЬ. Двоичные же файлы представляют собой последовательность любых символов.

С точки зрения содержания записанной в них информации файлы можно разделить на исполняемые (программы) и неисполняемые (файлы данных и документов). Исполняемые файлы могут запускаться операционной системой на выполнение, а неисполняемые файлы могут только изменять свое содержимое в процессе выполнения прикладных программ, с помощью которых они были созданы.

Кроме того, файлы можно разделить на:

- основные, наличие которых обязательно для работы как операционной системы, так и программ, работающих под ее управлением;
- служебные, хранящие конфигурацию и настройки основных файлов;
- рабочие, которые создаются и обрабатываются прикладными программами;
- временные файлы, создающиеся в момент работы прикладных программ и хранящие промежуточные результаты.

Следует отметить, что существует специальный вид файла, называемый папкой (ditectory) или каталогом (catalog). В таком файле содержатся ссылки на другие файлы. Поскольку ссылки на эти файлы содержатся лишь в одной из папок, для пользователя эти файлы как бы расположены в этой папке. На самом деле, конечно, все файлы находятся в секторах диска. Но это истинно лишь на физическом уровне, а на уровне представления данных файлы находятся в папках. Папки используются для упорядочения программ и документов на диске и могут вмещать как файлы, так и другие папки. Папки появились не сразу и не во всех операционных системах. Они возникли там, где требовался большой объем хранимой информации (например, в файловых системах жестких дисков) и, следовательно, возникли сложности с

организацией и размещением большого числа файлов.

Любой файл имеет служебные поля, в которых записывается информация о нем. Она включает имя, дату и время создания и модификации, свой размер и атрибуты: только для чтения — **Read only;** скрытый файл — **Hidden;** системный файл — **System;** архивный файл — **Archive.**

То же относится и к папкам. Имя и тип любого файла (и папки) должны быть уникальными в пределах той папки, в которой они находятся. Операционные системы предъявляют определенные требования к именам, которые присваиваются файлам.

41. Компьютерные сети и их назначение.

Компьютерной сетью называется группа компьютеров, объединенных линиями передачи данных и способных обмениваться информацией. Компьютеры могут располагаться в одной или в рнчличных фирмах либо в различных географических точках. Компьютерные сети следует отличать от многомашинных комплексов. Многомашинным вычислительным комплексом называется несколько компьютеров, соединенных средствами сопряжения и играющих разную роль в едином вычислительном процессе.

Всемирная тенденция к объединению компьютеров в сети обусловлена рядом важных объективных причин, таких, как глобализация экономики, повышение уровня управления предпринимательскими и государственными структурами, появление новых видов информационных услуг. Пользователи, подключенные к компьютерной сети, могут получать и передавать сообщения по электронной почте, имеют доступ к информации вне зависимости от ее географического расположения, а также возможность пользоваться программным обеспечением различных фирм.

Без использования компьютерных сетей невозможно создать информационную систему, эффективно управляющую предприятием.

42. Классификация вычислительных сетей.

Все вычислительные сети можно классифицировать по ряду признаков. В зависимости от расстояний между ПК различают следующие вычислительные сети:

1. локальные вычислительные сети – ЛВС (LAN – Local Area Networks) – компьютерные сети, расположенные в пределах небольшой ограниченной территории (здании или в соседних зданиях) не более 10 – 15 км;
2. территориальные вычислительные сети, которые охватывают значительное географическое пространство. К территориальным сетям можно отнести городские (MAN - Metropolitan Area Network),
3. региональные (Regional computer network), национальные (National computer network) и глобальные (WAN - Wide Area Network) сети. Городские и

региональные сети связывают абонентов района, города или области. Глобальные сети объединяют абонентов, удаленных между собой на значительное расстояние, находящихся в различных странах или континентах.

43. Сеть моноканальной топологии (Технология Етнернет).

При организации взаимодействия узлов в локальных сетях Основную роль играет используемый метод доступа к каналу связи. По приведенной классификации доступ к каналу связи обеспечивает канальный уровень. Большинство локальных сетей содержат несколько десятков компьютеров и территориально ограничены. Главное требование к ним — надежность, простота управления и приемлемая цена. Еще при возникновении локальных сетей перед разработчиками встала задача нахождения простого и дешевого решения для объединения компьютеров в вычислительную сеть. В результате было предложено несколько схем кабельных соединений между компьютерами. Три разные идеи были использованы в наиболее известных технологиях Токеп Ring, Arcnet и Ethernet.

Спецификацию Ethernet в конце 1970-х годов предложила компания Xerox Corporation. Сообщение, отправляемое одним узлом, принимается одновременно всеми остальными узлами, подключенными к общей шине. Но сообщение предназначено только для одного из них (оно включает в себя адрес узла назначения м адрес отправителя). Тот узел, которому предназначено сообщение, примет его, остальные его пропустят.

Метод доступа Ethernet является методом множественного доступа с прослушиванием канала связи. Перед началом передачи рабочая станция определяет, свободен канал или занят. Если кпиал свободен, станция начинает передачу. Ethernet исключает нозможности одновременной передачи сообщений двумя или неекольким станциями.

Данная топология применяется в локальных сетях с архитектурой Ethernet (классы 10Base-5 и 10Base-2 для толстого и тонкого коаксиального кабеля соответственно).

Преимущества сетей шинной топологии:

- отказ одного из узлов не влияет на работу сети в целом;
- сеть легко настраивать и конфигурировать;
- сеть устойчива к неисправностям отдельных узлов.

Недостатки сетей шинной топологии:

- разрыв кабеля может повлиять на работу всей сети;
- ограниченная длина кабеля и количество рабочих станций;
- трудно определить дефекты соединений

44. Сеть кольцевой топологии. (Технология Токен Ring).

При топологии«кольцо» компьютеры подключаются к кабелю, замкнутому в кольцо. Сигналы передаются по кольцу в одном направлении и проходят

через каждый компьютер. В отличие от пассивной топологии «шина», здесь каждый компьютер выступает в роли повторителя, усиливая сигналы и передавая их следующему компьютеру. Поэтому, если выйдет из строя один компьютер, прекращает функционировать вся сеть. Следовательно, трудно локализовать проблемы, а изменение конфигурации требует остановки всей сети. Оборудование для сетей с топологией кольцо более дорогостоящее.

К преимуществам можно отнести: устойчивость сети к перегрузкам (нет коллизий, отсутствует центральный узел) и возможность охвата большой территории. Кроме того, количество пользователей не оказывает большого влияния на производительность сети.

Метод доступа, используемый в технологии Token Ring, предусматривает следующие правила:

- все устройства, подключенные к сети, могут передавать дам ные, только получив разрешение на передачу (маркер);
- в любой момент времени только одна станция в сети обладне і таким правом;
- данные, передаваемые одной станцией, доступны всем стнм циям сети.

45. Сеть звездообразной топологии (Технология Arcnet).

Технология Arcnet (Attached Resource Computer Network) — простая, недорогая, надежная и гибкая архитектура локальной сети, разработанная корпорацией *Datapoints* 1977 г. По технологии *Arcnel* один из компьютеров создает специальный маркер (сообщение специального вида), который последовательно передается от од ного компьютера к другому. Если станция желает передать сообщение другой станции, она дожидается маркера, добавляет к нему сообщение, дополненное адресами отправителя и получателя со общения, и посылает его дальше; следующий узел также может присоединить к группе свое сообщение и т.д. В результате по ней и проходит поток из нескольких сообщений, возглавляемых кольцевым маркером. Компьютер, которому адресовано одно из сообщений потока, отцепляет его. Технология *Arcnet* может исполь зоваться при любой топологии.

Метод доступа, используемый в технологии *Arcnet,* предусматривает следующие правила:

- устройства, подключенные к сети, могут передавать данный по очереди, получив разрешение на передачу (маркер специ ального вида);
- в любой момент времени передавать данные может только один станция в сети;
- данные, передаваемые одной станцией, доступны всем станциям сети.

Передача каждого байта в *Arcnet* выполняется специальной но сылкой, состоящей из 3 служебных старт/стоповых битом и 8 битов данных. В

начале каждого пакета передается начальный разделитель, который состоит из 6 служебных битов. Прежде чем посылать информационное сообщение, отправитель посылает запрос о готовности к приему данных. После приема сообщении посылается извещение о факте получения и об отсутствии ошибок в пакете.

46. Структура сети Интернет.

Глобальные сети предназначены для максимально широкого обмена и распространения информации, так как они связывают абонентов в пределах целой страны, континента или всего земного шара. Фактически, глобальная сеть - это множество компьютеров, обменивающихся между собой информацией преимущественно посредством телефонных линий или спутников связи.
Узлы и магистрали сети Интернет - это ее инфраструктура, а в сети Интернет существует несколько сервисов или служб (E-mail, USENET, TELNET, WWW, FTP и др.), одним из первых сервисов является электронная почта E-mail. В настоящее время большая часть трафика в Интернет приходится на службу WorldWideWeb(всемирная паутина).

Принцип работы сервиса WWW был разработан физиками Тимом Бернес-Ли и Робертом Кайо в европейском исследовательском центре CERN (Женева) в 1989 году. В настоящее время Web – служба Интернет содержит миллионы страниц информации с различными видами документов. Компоненты структуры сети Интернет объединяются в общую иерархию. Интернет объединяет множество различных компьютерных сетей и отдельных компьютеров, которые обмениваются между собой информацией. Вся информация в Интернет хранится на Web-серверах. Обмен информацией между Web-серверами осуществляется по высокоскоростным магистралям. К таким магистралям относятся: выделенные телефонные аналоговые и цифровые линии, оптические каналы связи и радиоканалы, в том числе спутниковые линии связи. Серверы, объединенные высокоскоростными магистралями, составляют базовую часть Интернет. Пользователи подключаются к сети через маршрутизаторы местных поставщиков услуг Интернета или провайдеров (ISP), которые имеют постоянное подключение к Интернет через региональных провайдеров. Региональный провайдер, подключается к более крупному провайдеру национального масштаба, имеющего узлы в различных городах страны.

47. Адресация ресурсов в интернете. Доменная система имен.

Структура Интернет напоминает паутину, в узлах которой находятся компьютеры, связанные между собой линиями связи. Узлы Интернет, связанные высокоскоростными линиями связи, составляют базис Интернет. Как правило, это поставщики услуг (провайдеры). Оцифрованные данные

пересылаются через маршрутизаторы, которые соединяют сети с помощью сложных алгоритмов, выбирая маршруты для информационных потоков.

Каждый компьютер в Интернет имеет свой уникальный адрес. В протоколе TCP/IP каждый компьютер адресуется четырьмя отделяемыми друг от друга точками десятичными числами, каждое из которых может иметь значение от 1 до 255. Адрес компьютера выглядит следующим образом:

19.226.192.108

Такой адрес называется IP-адресом. Пользователю неудобно запоминать такие адреса, которые к тому же могут изменяться. Поэтому в Интернет существует Доменная Служба Имен (DNS - Domain Name System), которая позволяет каждый компьютер назвать по имени. В сети существуют миллионы компьютеров, и чтобы имена не повторялись, они разделены по независимым доменам.

Таким образом адрес компьютера выглядит как несколько доменов, разделенных точкой:

<сегмент n>. … <сегмент 3>.<сегмент 2>.<сегмент 1>.

Здесь сегмент 1 – домен 1 уровня, сегмент 2 – домен 2 уровня и т.д.

Доменное имя - это уникальное имя, которое данный поставщик услуг избрал себе для идентификации, например: ic.vrn.ru или yahoo.com

Домен 1 уровня обычно определяет страну местоположения сервера (ru – Россия; ua – Украина; uk – Великобритания; de – Германия) или вид организации (com – коммерческие организации; edu - научные и учебные организации; gov - правительственные учреждения; org – некоммерческие организации).

Когда вводится доменное имя, например, www.mrsu.ru, компьютер должен преобразовать его в адрес. Чтобы это сделать, компьютер посылает запрос серверу DNS, начиная с правой части доменного имени и двигаясь влево. Его программное обеспечение знает, как связаться с корневым сервером, на котором хранятся адреса серверов имён домена первого уровня (крайней правой части имени, например, ru). Таким образом, сервер запрашивает у корневого сервера адрес компьютера, отвечающего за домен ru. Получив информацию, он связывается с этим компьютером и запрашивает у него адрес сервера mrsu. После этого от сервера mrsu он получает адрес www компьютера, который и был целью данной прикладной программы.

48. Всемирная паутина (www).

Сеть WWW образуют миллионы *веб-серверов*, расположенных по всему миру. *Веб-сервер* является программой, запускаемой на подключенном к сети компьютере и передающей данные по протоколу HTTP.

Для идентификации ресурсов (зачастую файлов или их частей) в WWW используются идентификаторы ресурсов *URI* (Uniform Resource Identifier). Для определения местонахождения ресурсов в этой сети используются

локаторы ресурсов *URL* (Uniform Resource Locator). Такие URL-локаторы представляют собой комбинацию URI и системы DNS.
Доменное имя (или IP-адрес) входит в состав URL для обозначения компьютера (его сетевого интерфейса), на котором работает программа веб-сервер.
На клиентском компьютере для просмотра информации, полученной от веб-сервера, применяется специальная программа — *веб-браузер*.
Основная функция веб-браузера - отображение гипертекстовых страниц (веб-страниц). Для создания гипертекстовых страниц в WWW изначально использовался язык HTML. Множество веб-страниц образуют *веб-сайт*.
Разработкой стандартов для сети Веб, начиная с 1994 года, занимается Консорциум W3C (World Wide Web Consortium), основанный и до сих пор возглавляемый Тимом Бернерсом-Ли.
Консорциум W3C — организация, разрабатывающая и внедряющая технологические стандарты для Интернета и WWW. Миссия W3C формулируется следующим образом: "Полностью раскрыть потенциал Всемирной паутины путем создания протоколов и принципов, гарантирующих долгосрочное развитие Сети". Две другие важнейшие задачи Консорциума — обеспечить полную "интернационализацию Сети" и сделать ее доступной для людей с ограниченными возможностями.
W3C разрабатывает для WWW единые принципы и стандарты, называемые "*Рекомендациями*", которые затем внедряются разработчиками программ и оборудования. Благодаря *Рекомендациям* достигается совместимость между программными продуктами и оборудованием различных компаний, что делает сеть WWW более совершенной, универсальной и удобной в использовании.

49. База данных. Классификация и элементы базы данных.

База данных(БД) – это поименованная совокупность структурированных данных, относящихся к определенной предметной области
СУБД – это комплекс программных и языковых средств, необходимых для создания баз данных, поддержаниия их в актуальном состоянии и организация поиска в них необходимой информации
По технологии обработки БД подразделяются на централизованные и распределенные. Централизованные БД хранится в памяти одной вычислительной системы. Если эта вычислительная система является компонентом сети ЭВМ, возможен распределенный доступ к такой базе.
Распределенная БД состоит из нескольких, возможных пересекающихся или даже дублирующий друг друга частей, хранимых в различных ЭВМ.
Работа с такой базой осуществляется с помощью системы управления распределенной базой данных(СУРБД)
Понятие БД связано с такими понятиями структурных элементов, как поле , запись, файл.

Поле – это элементарная единица логической организации данных, которая соответствует неделимой единицы информации- реквизиту.
Для описания поля используется следующие характеристики
Имя , например, Фамилия , Имя, Отчество. Дата рождения
Тип, например. 15 байт. Причем будет определяться максимально возможным количеством символов.
Запись – это совокупность логически связанных полей.
Экземпляр записи – это отдельная реализация записи, содержащая конкретные значения ее полей
Файл(таблица) – это совокупность экземпляров записей одной структуры

50. Реляционная модель данных. Структуры данных реляционной модели

Понятие реляционный (англ. Relation- отношение) связано с разработками известного американского специалиста в области систем без данных Е.Кодда.
Эти модели характеризуются простотой структуры данных, удобным для пользователя табличнвм представлением и возможностью использования алгебры отношений и реляционного исчисления для обработки данных.
Реляционная модель ориентирована на организацию данных в виде двумерных таблиц.Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

4. Каждый элемент таблицы-один элемент данных ;
5. Все столбцы в таблицах однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой,символьный и т.д.) и длину;
6. Каждый столбец имеет уникальное имя;
7. Одинаковые строки в таблице отсутствуют;
8. Порядок следования строк и столбцов может быть произвольным.