Филиал федерального государственного бюджетного образовательного учреждения высшего образования
«Национальный исследовательский университет «МЭИ»
в г. Смоленске

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБЕСПЕЧЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА

Направление подготовки: 13.03.01 «Теплоэнергетика и теплотехника»

Профиль: «Энергообеспечение предприятий»

Уровень высшего образования: бакалавриат

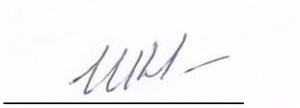
Нормативный срок обучения: 4 года

Форма обучения: очная

Год набора: 2020

Смоленск

Методические материалы составил:



_____ _к.т.н., доцент_ _Кабанова И.А._
подпись ФИО

«_24_» _июня_ 2020г.

Заведующий кафедрой «Промышленная теплоэнергетика»:

_____ _В.А. Галковский_
подпись Ф.И.О.

«_02_» _июля_ 2020 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические указания для обучающихся по освоению дисциплины «Информационные технологии» разработаны для подготовки студентов по направлению 13.03.01 «Теплоэнергетика и теплотехника» профиль «Энергообеспечение предприятий» в соответствии с рабочей программой дисциплины «Информационные технологии» и включают:

Учебно-методическое обеспечение лекционных занятий

Методические рекомендации по выполнению практических занятий

Методические рекомендации по выполнению лабораторных работ

Методические рекомендации к самостоятельной работе студентов по дисциплине «Тепломассообмен».

Филиал федерального государственного бюджетного образовательного учреждения высшего образования
«Национальный исследовательский университет «МЭИ»
в г. Смоленске

Учебно-методическое обеспечение лекционных занятий
по дисциплине

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

(НАИМЕНОВАНИЕ ДИСЦИПЛИНЫ)

Смоленск

1. ВВЕДЕНИЕ В ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

1.1 Понятие информационной технологии

Термин *технология* произошел от греческого teche + logos (мастерство + учение). В производственном процессе под технологией понимают систему взаимосвязанных способов обработки материалов и приемов изготовления продукции. В общем случае технология – это правила действия с использованием каких-либо средств, которые являются общими для целой совокупности задач или задачных ситуаций. Если реализация технологии направлена на выработку управляющих воздействий, то это *технология управления.*

Информационные процессы (сбор, обработка и передача информации) всегда играли важную роль в науке, технике и жизни общества. В ходе эволюции человечества просматривается устойчивая тенденция к автоматизации этих процессов, хотя их внутреннее со- держание по существу часто остается неизменным.

Таким образом, АИТ — это ИТ, в которой для всех операций, связанных с информацией, используются средства автоматизации. Далее в настоящем учебнике под ИТ будут пониматься именно автоматизированные информационные технологии.

Определение информационных технологий как сферы человеческой деятельности предложено ЮНЕСКО: информационные технологии — это комплекс взаимосвязанных научных, технологических, инженерных дисциплин, изучающих методы эффективной организации труда людей, занятых обработкой и хранением информации; вычислительную технику и методы организации и взаимодействия с людьми и производственным оборудованием, их практические приложения, а также связанные со всем этим социальные, экономические и культурные проблемы.

Чтобы терминологически выделить традиционную технологию решения экономических и управленческих задач, был введен термин «предметная технология».

Так, любой участок бухгалтерского учета предполагает поступление первичной документации, которая трансформируется в форму бухгалтерской проводки. Последняя, изменяя состояние аналитического учета, приводит к изменению счетов синтетического учета и далее — баланса. Предметная технология и информационная технология влияют друг на друга. Например, использование компьютера для операций по бухгалтерскому учету внесло изменения в предметную технологию, исключив из обработки значительное количество журналов и ведомостей, используемых при традиционной системе учета, а также предоставив принципиально новые оперативные возможности. С другой стороны, предметные технологии, наполняя специфическим содержанием информационные технологии, акцентируют их на вполне определенные функции.

Примеры обеспечивающих технологий: технологии баз данных, технология клиент-сервер, технологии обработки текстов, технологии Data Mining и т.д.

Так, например, работа юриста, использующего ПК, обязательно предполагает применение следующих технологий:

- технологии автоматизированного составления документов (договоров, исковых заявлений и т.д.) на основе типовых шаблонов;
- технологий поиска документированной информации (реализованных в справочно-правовых системах);
- использование алгоритмов юридических расчетов (расчет государственной пошлины, процентов по задолженности, пени);
- технологии бухгалтерского учета (например, учет рабочего времени, использующийся в деятельности частнопрактикующих юристов с повременной оплатой);
- использование других технологий, реализованных в какой либо ИТ (СУБД, текстовом процессоре, электронной таблице).

Трансформация обеспечивающей ИТ в функциональную (модификация некоторого общеупотребительного инструментария в специальный) может быть сделана как специа-

листом-проектировщиком, так и самим пользователем, в зависимости от того, насколько она сложна (т.е. доступна самому пользователю).

Таким образом, функциональная технология представляет собой синтез обеспечивающей и предметной технологий, осуществленный по некоторым правилам. Являясь некоторой средой преобразования данных и одновременно частью информационной системы, она базируется на платформе, которая состоит из технической, программной, организационной и информационной части.

Понятия АИС и ИТ являются взаимопроникающими и взаимодополняющими. С одной стороны ИТ базируется на АИС; можно сказать, что информационная технология — это информационная система плюс действия персонала по работе с нею, направленные на получение нужного результата. С другой стороны (эта точка зрения закреплена в российском законодательстве) информационная система может пониматься как «совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств». Очевидно, что рассматривать информационные системы и информационные технологии в отрыве друг от друга бессмысленно. Одна- ко при характеристике АИС мы в первую очередь акцентируем внимание на ее структуре и используемых алгоритмах, в то время, как говоря об ИТ, основное внимание следует уделять этапам технологического процесса, а также аспектам взаимодействия пользовате- ля с программно-техническими средсвами и программно-технических средств друг с дру- гом на этих этапах.

1.2 Составляющие информационных технологий

Выделяют следующие основные составляющие ИТ (одинаково характерные и для автоматизированных информационных систем):

1. *Техническое обеспечение* — это аппаратные средства и средства коммуникациии, обеспечивающие работу ИТ. Как правило, включают персональный компьютер, периферийные устройства, линии связи, сетевое оборудование и т.д. От технического обеспечения зависит главным образом характер сбора и передачи данных.
2. *Программное обеспечение* (ПО) непосредственно реализует функции накопления, обработки, хранения, отображения, поиска и анализа данных, обеспечивает взаимодействие пользователя с ЭВМ посредством пользовательского интерфейса. ПО находится в прямой зависимости от технического обеспечения.
3. *Информационное обеспечение* представляет собой совокупность проектных решений по видам, объемам, способам размещения и формам организации информации, циркулирующей в ИС.
4. *Методическое обеспечение* — это комплекс нормативно-методических и инструктивных материалов подготовки и оформления документов по эксплуатации технических средств, организации работы специалистов-пользователей и технического персонала. *Организационное обеспечение* представляет собой комплекс методов, средств и документов, регламентирующих взаимодействие персонала информационной системы с техническими средствами и между собой в процессе разработки и эксплуатации информационной системы. Таким образом, организационное и методическое обеспечение увязывают действия персонала по работе с техническими и программными средствами в единый технологический процесс.
5. *Математическое обеспечение* — это совокупность математических методов, моделей и алгоритмов обработки информации, используемых при решении функциональных задач и в процессе автоматизации проектировочных работ автоматизированной информационной технологии. Математическое обеспечение включает средства моделирования процессов управления, методы и средства решения типовых задач управления, методы оптимизации запасов материальных ресурсов и принятия оптимальных управленческих решений.

6. *Правовое обеспечение* — собой совокупность правовых норм, регламентирующих правоотношения при создании, внедрении и использовании ИТ.

7. *Лингвистическое обеспечение* включает совокупность научно-технических терминов и других языковых средств, используемых в ИТ, а также правил формализации естественного языка, включающих методы сжатия и раскрытия текстовой информации с целью повышения эффективности автоматизированной обработки информации и облегчающих общение человека с ЭИС. Они делятся на две группы: традиционные языки (естественные, математические, языки программирования и моделирования) и языки, предназначенные для диалога с ЭВМ (информационно-поисковые языки, языки СУБД, языки операционных сред и т.д.)

1.3 Эволюция информационных технологий

Первая известная человечеству информационная технология — передача знаний устно по наследству. «Техническое обеспечение» этой технологии составляют язык и речь. С возникновением письменности появляется возможность *накопления* и *хранения* информации (что особенно важно, в неискаженном виде). Изобретение печатного станка и появление книгопечатания (а также более позднее развитие фотографии и звукозаписи) позволяют усовершенствовать эти процессы.

Сигнализация (например, с помощью дыма костров) может считаться одной из первых *коммуникационных* технологий (технологий *передачи* информации на расстоянии). Затем ей на смену последовательно приходят почта, телеграф, телефон, радио, телевидение.

Технологии *обработки* информации до середины XX века полностью сводились к подсчету и вычислениям (с помощью камешков, абака, счетов, механического арифмометра и логарифмической линейки).

Автоматизированные информационные технологии появляются после 1949 года, когда была построена первая ЭВМ. Выделяют следующие основные этапы их развития:

I этап — до начала 60-х годов — время ЭВМ первого и второго поколений. Основным требованием к информационным технологиям является экономия дорогих машинных ресурсов. Широко применяется технология пакетной обработки данных. Задания (представляющие собой данные и программы для их обработки) объединяются в пакет, который и отправляется на обработку. Распределение ресурсов (в частности, процессорного времени) между заданиями происходит таким образом, чтобы максимизировать загрузку оборудования. Другие характерные черты этого этапа: программирование в машинных кодах, появление блок-схем, разработка библиотек стандартных программ. Был разработан операторный метод, который послужил основой для разработки алгоритмических языков (Алгол, Кобол, Фортран).

II этап — до начала 80-х годов — начинается с появлением мини-ЭВМ на больших интегральных схемах. Основным требованием к ИТ становится экономия труда программиста. Появляются инструментальные средства программирования, такие как языки высокого уровня (Pascal , C + и др.), системы управления базами данных (СУБД), системы автоматизации проектирования (САПР), диалоговые средства общения с ЭВМ, новые технологии проектирования (структурное и модульное). Появляются операционные системы второго поколения, которые помимо пакетного могут работать в режимах реального времени и разделения времени. С появлением глобальных сетей начинают развиваться коммуникационные технологии (электронная почта, поисковые системы и др.) Совокупность научных методов и технологических приёмов, ориентированных на обработку данных, стали называть *информатикой*.

III этап — до начала 90-х годов — начинается с появлением персонального компьютера (ПК). Информационные технологии начинают использоваться для автоматизации процессов из всех сфер человеческой деятельности. В связи с этим особенно актуальной

стоит задача формализации этих процессов. Большое внимание уделяется совершенствованию пользовательского интерфейса. Получают широкое распространение диалоговые операционные системы, автоматизированные рабочие места (АРМ), экспертные системы, базы знаний, локальные вычислительные сети, распределённая обработка данных.

С повсеместным внедрением ПК связывают четвертую информационную революцию. Первой информационной революцией принято считать изобретение письменности, второй — книгопечатания, третьей — телефона и телеграфа. *Информационная революция* — преобразование общественных отношений из-за кардинальных изменений в сфере обработки информации.

IVэтап развития ИТ – 90-е годы. Информация становится стратегическим ресурсом. Новые информационные технологии преследуют цель информатизации общества. Формируются базы знаний по всем областям человеческой деятельности. Появляются портативные ЭВМ, операционные системы с графическим интерфейсом, объектно- ориентированные, гипертекстовые, мультимедийные технологии. Средства электронной коммуникации (электронная почта, ICQ) становятся повседневным инструментом обще- ния.

Эволюция информационных технологий неразрывно связана с развитием человеческого общества. Являясь являются продуктом развития производственных и общественных отношений, информационные технологии становятся одновременно и катализатором, ускоряющим процесс развития человеческого общества.

1.4 Роль ИТ в развитии экономики и общества

Бурное развитие компьютерной техники и ИТ послужило толчком к развитию *информационного общества*.

В информационном обществе изменятся не только производство, но и весь уклад жизни, система ценностей, возрастет значимость культурного досуга по отношению к материальным ценностям. По сравнению с индустриальным обществом, где все силы направлены на производство и потребление товаров, в информационном обществе производятся и потребляются в основном интеллект и знания, что приводит к увеличению доли умственного труда.

Основные признаки информационного общества:

— приоритет информации по сравнению с другими ресурсами;

— глобальный характер информационных технологий: они охватывают все сферы социальной деятельности человека;

— реализован свободный доступ каждого человека к информационным ресурсам всей цивилизации.

Проблемы информационного общества:

— возрастающее влияние на общество средств массовой информации;

— возможности ИТ создают серьезную угрозу тайне частной жизни людей;

— сложность отбора качественной и достоверной информации;

— трудности в адаптации части людей к среде информационного общества.

Ближе всех на пути к информационному обществу стоят страны с развитой информационной индустрией: США, Япония, Англия, Германия, страны Западной Европы.

В информационном обществе от человека требуется умение быстро воспринимать и обрабатывать большие объемы информации, владение современными методами работы с ней. Эти качества составляют основу информационной культуры.

Наибольшее влияние на формирование информационного общества оказывают ИТ массового применения, воздействующие не только на развитие культуры общества, экономики и производства, но и на социальный уклад жизни людей и стереотипы их поведения. Использование ИТ позволяет не только изучать и предсказывать различные перемены в общественной жизни, но и активно влиять на практическую политическую, экономиче-

скую, культурную жизнь, получать прогнозируемый результат. Все шире утверждается инновационный метод освоения социального пространства — его технологизация.

1.5 Свойства информационной технологии

Информационные технологии характеризуются следующими свойствами:

1. *Целесообразность*. Внедрение ИТ не является самоцелью. ИТ реализуется с целью повышения эффективности производства (или иной деятельности) путем автоматизации тех бизнес-процессов, в которых большую роль играет надежность, своевременность и полнота информации.
2. *Системная полнота* или *целостность* процесса. Процесс включает все элементы, обеспечивающие необходимую завершенность действий человека в достижении поставленной цели.
3. *Регулярность* процесса и *высокая степень расчлененности* его на однозначные фазы (стадии, этапы). *Однозначность фаз* позволяет применять средние величины при их характеристике, а значит, осуществить стандартизацию и унификацию.
4. *Взаимодействие с внешней средой*.
5. *Реализация во времени*. Информационная технология должна соответствовать новым потребностям организации, реагировать на изменения бизнес-процессов и учитывать новые возможности технических средств и программного обеспечения. Поэтому после внедрения ИТ, как правило, происходит ее постоянное развитие (модификация, изменение структуры, включение новых компонентов).

1.6 Структура информационной технологии

Как отмечалось выше, ИТ внедряется для автоматизации определенных процессов циркуляции и переработки информации в некоторой предметной области. Поэтому на реализацию информационной технологии (в частности, выбор аппаратного, программного, организационного обеспечения) влияют, в первую очередь, содержание этих процессов и особенности предметной области. Обычно работа над внедрением ИТ начинается с создания моделей предметной области.

ИТ строится на базе некоторой *опорной технологии,* которая включает:

•аппаратные средства;

•системное ПО (операционные системы, драйверы, системные утилиты);

•инструментальное ПО (системы управления базами данных, средства разработки программного обеспечения).

Ядром информационной технологии являются три компонента.

1. *База данных*, посредством которой реализуется функция хранения всей информации, необходимой для эффективной реализации бизнес-процесса. Вместо классических реляционных БД могут использоваться другие варианты. Например, большинство технологий Data Mining строятся на основе *хранилища данных,* а экспертные системы обращаются к *базе знаний*.
2. *Пользовательский интерфейс,* посредством которого осуществляется взаимодействие между персоналом и информационной системой (такое взаимодействие является необходимым для многих этапов технологического процесса). Современные ИТ используют концепцию автоматизированных рабочих мест, в соответствии с которым каждый сотрудник взаимодействует с АИС непосредственно на своем рабочем месте.
3. Совокупность *технологических процессов*, реализуемых посредством данной ИТ.

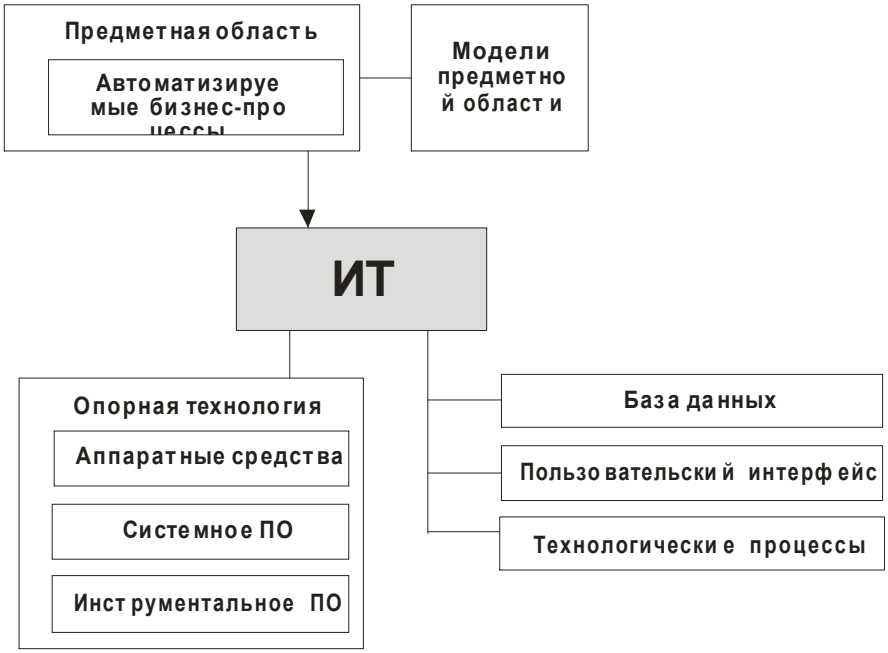


Рис. 1.1. Структура информационной технологии

1.7 Классификация информационных технологий

Классификация информационных технологий зависит от критерия классификации. В качестве критерия может выступать показатель или совокупность признаков, влияющих на выбор той или иной информационной технологии.

На ранних этапах развития ИТ[5] широко использовалось их разделение *по типу обрабатываемых данных:* текстовые, табличные, графические, гипертекстовые и т.д. В настоящее время такая классификация неактуальна, хотя некоторые классы ИТ можно выделить именно по этому признаку.

Геоинформационные технологии предназначены для решения научных и прикладных задач инвентаризации, анализа, оценки, прогноза и управления окружающей средой и территориальной организацией общества.

Классифицируя информационную технологию *по типу носителя информации*, говорят о *бумажной* и *безбумажной* технологиях.

По степени типизации операций выделяют *пооперационные* и *попредметные* технологии. В первом случае за каждой операцией закрепляется рабочее место с техническим средством. Попредметная технология подразумевает выполнение совокупности взаимосвязанных операций (например, операций по ведению бухгалтерского учета) на одном рабочем месте. Такой технологии соответствует концепция АРМ.

По степени распределенности выполняемых операций выделяют *локальные* и *сетевые (распределенные)* информационные технологии.

В настоящее время наиболее удобной классификацией информационных технологий является классификация *по степени автоматизации задач управления*. В соответствии с этим признаком выделяют пять основных видов ИТ:

1. Технологии обработки данных. Предназначены для решения хорошо структурированных задач, алгоритмы решения которых хорошо известны и для решения которых имеются все необходимые входные данные. Эта технология применяется на уровне ис-

полнительской деятельности персонала невысокой квалификации в целях автоматизации некоторых рутинных, постоянно повторяющихся операций управленческого труда.

2. Автоматизация функций управления. Цель такой ИТ — удовлетворение информационных потребностей всех без исключения сотрудников фирмы, имеющих дело с принятием решений, на основе различных видов отчетов.

3. Электронный офис. Обеспечивает организацию и поддержку коммуникационных процессов как внутри организации, так и с внешней средой на базе компьютерных сетей и других современных средств передачи и обработки информации. Технология автоматизации офиса строится на базе таких продуктов как текстовые и табличные редакторы, электронная почта, электронный календарь, телеконференции, системы электронного документооборота и т.д.

4. ИТ поддержки принятия решений. Помогает человеку перерабатывать большие объемы информации и принимать решения. Особенность данной ИТ в том, что человек участвует в данном процессе на начальной и завершающей стадиях (вводит данные в компьютер и принимает окончательное решение на основе полученной информации), а компьютер под управлением человека создает новую информацию.

5. Экспертная поддержка. Дает возможность получать консультации экспертов по любым проблемам, о которых этими системами накоплены знания. В отличие от систем поддержки принятия решений экспертные системы могут предложить решения, превосходящие интеллектуальные возможности пользователя.

1.8 Безопасность информационных технологий

В связи с повсеместным использованием сетевых информационных технологий, особенную актуальность в настоящее время приобретает разработка защищенных ИТ, обеспечение безопасности данных.

Под *информационной безопасностью* понимают состояние защищенности информации и информационной среды от случайных или преднамеренных воздействий естественного или искусственного характера, которые могут нанести неприемлемый ущерб субъектам информационных отношений, (в том числе владельцам и пользователям информации). *Защита информации* – комплекс мероприятий, направленных на обеспечение информационной безопасности.

Самая распространенная модель информационной безопасности базируется на обеспечении трех свойств информации: конфиденциальность, целостность и доступность.

Конфиденциальность информации означает, что с ней может ознакомиться только строго ограниченный круг лиц, определенный ее владельцем. Если доступ к информации получает неуполномоченное лицо, происходит утрата конфиденциальности.

Для некоторых типов информации конфиденциальность является одним из наиболее важных атрибутов (например, данные стратегических исследований, медицинские и страховые записи, спецификации новых изделий и т. п.). В определенных случаях важно сохранить конфиденциальность сведений о конкретных лицах (например, сведения о клиен- тах банка, о кредиторах, налоговые данные; сведения медицинских учреждений о состоя- нии здоровья пациентов и т. д.).

Целостность информации определяется ее способностью сохраняться в неискаженном виде. Неправомочные, и не предусмотренные владельцем изменения информации (в результате ошибки оператора или преднамеренного действия неуполномоченного лица) приводят к потере целостности. Целостность особенно важна для данных, связанных с

функционированием объектов критических инфраструктур (например, управления воздушным движением, энергоснабжения и т. д.), финансовых данных.

Доступность информации определяется способностью системы предоставлять своевременный беспрепятственный доступ к информации субъектам, обладающим соответствующими полномочиями. Уничтожение или блокирование информации (в результате ошибки или преднамеренного действия) приводит к потере доступности.

Доступность — важный атрибут для функционирования информационных систем, ориентированных на обслуживание клиентов (системы продажи железнодорожных билетов, распространения обновлений программного обеспечения). Ситуацию, когда уполномоченный пользователь не может получить доступ к определенным услугам (чаще всего сетевым), называют *отказом в обслуживании*.

Кроме перечисленых трех свойств дополнительно выделяют еще два свойства, важных для информационной безопасности: аутентичность и апеллируемость.

Аутентичность — возможность достоверно установить автора сообщения.

Апеллируемость — возможность доказать, что автором является именно данный человек и никто другой.

Угроза – потенциально возможное событие, действие, процесс или явление, которое может привести к нанесению ущерба чьим-либо интересам.

Соответственно *угрозой информационной безопасности* называется потенциально возможное событие, процесс или явление, которое посредством воздействия на информацию или компоненты АИС может прямо или косвенно привести к нанесению ущерба интересам субъектов информационных отношений.

Атака — попытка реализации угрозы.

Нарушение — реализация угрозы.

Наиболее распространенные угрозы информационной безопасности — это сбои оборудования (кабельной системы, дисковых систем, серверов, рабочих станций и т.д.), поте- ри информации (из-за инфицирования компьютерными вирусами, неправильного хране- ния архивных данных, нарушений прав доступа к данным), некорректная работа пользователей и обслуживающего персонала.

Средства защиты информации условно можно разделить на три класса: средства физической защиты, программные средства и административные меры защиты.

1. *Средства физической защиты*

— *Физическое управление доступом*. Эти меры позволяют контролировать и при необходимости ограничивать вход и выход сотрудников и посетителей.

— *Противопожарные меры.*

— *Защита поддерживающей инфраструктуры* (к поддерживающей инфраструктуре можно отнести системы электро-, водо- и теплоснабжения, кондиционеры, средства коммуникаций).

— *Защита от перехвата данных*, который может осуществляться как путем визуального наблюдения в бинокль за монитором, так и анализом побочных электромагнитных излучений и наводок.

2. *Программные средства*

— *Идентификация* позволяет субъекту (пользователю или процессу, действующему от имени определенного пользователя) назвать себя (сообщить свое имя). Посредством *аутентификации* вторая сторона убеждается, что субъект действительно тот, за кого он себя выдает.

— *Средства управления доступом* позволяют специфицировать и контролировать

действия, которые субъекты (пользователи и процессы) могут выполнять над объектами (информацией и другими компьютерными ресурсами). Т.е. имеется набор субъектов и набор объектов и задача управления доступом состоит в том, чтобы для каждой пары (субъект, объект) определить множество допустимых операций. Сделать это можно с помощью *матрицы доступа*.

—*Протоколирование* (сбор и накопление информации о событиях, происходящих в информационной системе) и *аудит* (оперативно проводимый анализ накопленной информации).

—*Шифрование.* Мощнейшее средство обеспечения конфиденциальности и контроля целостности. Шифрование — это процесс преобразования исходного сообщения М (называемого *открытым текстом*) в форму М' (*зашифрованный текст* или *шифртекст*). При этом провести обратное преобразование М' в М возможно только обладая некоторой дополнительной информацией, называемой *ключом*. Кроме шифрования используется ряд других криптографических методов, основанных на математических преобразованиях информации с целью противодействия распространенных угрозам.

—*Экранирование* — средство разграничения доступа клиентов из одного множества информационных систем к серверам из другого множества информационных систем.

—*Антивирусные программы. Программы-детекторы* осуществляют поиск характерного для вируса кода (сигнатуры) в оперативной памяти и в файлах и при обнаружении выдают соответствующее сообщение. Программы-доктора или *фаги* также «лечат» их, т.е. возвращают зараженные файлы в исходное состояние. Среди фагов выделяют *полифаги,* т.е. программы-доктора, предназначенные для поиска и уничтожения большого количест- ва вирусов. *Ревизоры* запоминают исходное состояние объектов незараженной системы и периодически сравнивают текущее состояние с исходным. Программы-*фильтры* — резидентные (то есть постоянно работающие) программы, предназначенные для обнаружения при работе компьютера подозрительных действий, характерных для вирусов. *Вакцины* — резидентные программы, предотвращающие заражение файлов.

3.*Административные меры защиты*

—*Управление персоналом*. Включает в себя *принцип разделения обязанностей* (ни один человек не должен нарушить критически важный для организации процесс) и *принцип минимизации привилегий* (пользователям предоставляются лишь те права доступа, которые необходимы им для выполнения служебных обязанностей).

—*Поддержание работоспособности системы*. Это выполнение рутинных необходимых действий, таких как резервное копирование, документирование, управление носителями (учет CD-дисков, дискет) и т.д.

—*Политика безопасности*.

Политика безопасности организации — совокупность руководящих принципов, правил, процедур и практических приёмов в области безопасности, которыми руководствуется организация в своей деятельности. Разаработка политики безопасности включа- ет определение следующих основных моментов:

—какие данные и насколько серьезно необходимо защищать;

—кто и какой ущерб может нанести организации в информационном аспекте;

—основные риски и способы их уменьшения до приемлемой величины.

С практической точки зрения политику безопасности можно условно разделить на три уровня: верхний, средний и нижний.

К верхнему уровню относятся решения, затрагивающие организацию в целом (как правило, носят общий характер и исходят от руководства). Например, цели организации в

области информационной безопасности, программа работ в облести информационной безопасности (с назначением ответственных за ее реализацию).

К *среднему уровню* относятся вопросы, касающиеся отдельных аспектов информационной безопасности, но важные для различных систем, эксплуатируемых организацией (например, использование на работе персональных ноутбуков, установка непроверенного программного обеспечения, работа с Интернетом и т.д.).

Политика безопасности *нижнего уровня* касается конкретных сервисов и должна быть наиболее детальной. Часто правила достижения целей политики безопасности нижнего уровня заложены в эти сервисы на уровне реализации.

Основные принципы построения систем защиты АИС

1. *Простота механизма защиты.* Используемые средства защиты не должны требовать от пользователей специальных знаний или значительных дополнительных трудозатрат. Они должны быть интуитивно понятны и просты в использовании,.
2. *Системность.* При разработке системы защиты и вводе ее в эксплуатацию необходим учет всех взаимосвязанных, взаимодействующих и изменяющихся во времени элементов, условий и факторов, значимых для обеспечения безопасности. В частности, должны быть учтены все слабые места АИС, возможные цели и характер атак, возможность появления принципиально новых угроз безопасности.
3. *Комплексность.* Предполагает согласованное применение разнородных средств при построении целостной системы защиты, перекрывающей все существенные каналы реализации угроз и не содержащей слабых мест на стыках отдельных ее компонентов. Целесообразно строить эшелонированную систему защиты, обеспечивающую комплексную безопасность на разных уровнях (внешний уровень — физические средства, организационные и правовые меры; уровень ОС; прикладной уровень).
4. *Непрерывность.* Мероприятия по обеспечению информационной безопасности АИС должны осуществляться на протяжении всего ее жизненного цикла — начиная с этапов анализа и проектирования и заканчивая выводом системы из эксплуатации. При этом наилучший результат достигается, когда разработка системы защиты идет параллельно с разработкой самой защищаемой АИС. Не допускается также никаких перерывов в работе средств защиты.
5. *Разумная достаточность.* Один из основополагающих принципов информационной безопасности гласит: абсолютно надежная защита невозможна. Любой самый сложный механизм защиты может быть преодолен злоумышленником при затрате соответствующих средств и времени. *Система защиты считается достаточно надежной, если средства, которые необходимо затратить злоумышленнику на ее преодоление значительно превышают выгоду, которую он получит в случае успеха.* Иногда используется

обратный принцип: расходы на систему защиты (включая потребляемые ей системные ре- сурсы и неудобства, возникающие в связи с ее использованием) не должны превышать стоимость защищаемой информации.

6. *Гибкость.* Система защиты должна иметь возможность адаптироваться к меняющимся внешним условиям и требованиям.
7. *Открытость алгоритмов и механизмов защиты.* Система должна обеспечивать надежную защиту в предположении, что противнику известны все детали ее реализации. Или иными словами, защита не должна обеспечиваться за счет секретности структуры и алгоритмов системы защиты АИС.

2 Пользовательский интерфейс

2.1 Понятие и виды пользовательского интерфейса

Пользовательский интерфейс — это набор приемов и средств взаимодействия пользователя с приложением.

В настоящее время рассматривается три основных вида пользовательского интерфейса, реализуемого операционными системами и прикладными программами.

1. Командный интерфейс означает выдачу на экран приглашения для ввода команды. Программы, имеющие командный интерфейс, называют консольными. Такой интерфейс реализуют операционные системы MS DOS, Linux (без запуска графической оболочки).

2. WIMP-интерфейс — современный графический интерфейс, характерный для ОС Windows. Аббривеатура расшифровывается как Windows (окно), Image (изображение), Menu (меню), Pointer (указатель). Каждая программа запускается в собственной области экрана, называемой окно (Window); объекты, с которыми может взаимодействовать поль- зователь, представлены в виде изображений (Image); основные команды сгруппированы в меню (Menu); выбор команд и ввод данных осуществляется при помощи указателя мыши (Pointer). Для выбора одного из них используется указатель.

3. SILK — (Speech-речь, Image-изображение, Language-язык, Knowledge-знание). В данном интерфейсе при воспроизведении речевой команды происходит переход от одних поисковых изображений к другим, согласно семантическим связям. Предполагается, что этой модели будут пидерживаться интерфейсы будущего.

Необходимые свойства пользовательского интерфейса:

1.*Естественность*

Естественный интерфейс — такой, который не вынуждает пользователя существенно изменять привычные для него способы решения задачи. Это, в частности, означает, что сообщения и результаты, выдаваемые приложением, не должны требовать дополнительных пояснений. Целесообразно также сохранить систему обозначений и терминологию, используемые в данной предметной области.

2.*Согласованность.*

а) Согласованность в пределах продукта

Одна и та же команда должна выполнять одни и те же функции, где бы она не встретилась.

б) Согласованность в пределах рабочей среды

Поддерживая согласованность с интерфейсом, предоставляемым операционной системой (Windows), приложение может опираться на те знания и навыки пользователя, которые он получил ранее при работе с другими приложениями.

в) Согласованность в использовании метафор

Метафора — знакомое пользователю понятие или образ, помогающее обеспечить интуитивно понятный интерфейс (например, Корзина Windows). Если поведение некоторого программного объекта выходит за рамки того, что обычно подразумевается под соответствующей ему метафорой, у пользователя могут возникнуть трудности при работе с таким объектом. (Например, если у программного объекта Корзина определить операцию Запуск, пользователь не поймет предназначения объекта без посторонней помощи).

3.*Дружественность интерфейса (Принцип «прощения» пользователя).*

Эффективный интерфейс должен принимать во внимание, что большинство пользователей изучают особенности работы с новым программным обеспечением методом проб и

ошибок. На каждом этапе работы интерфейс должен разрешать только соответствую- щий набор действий и предупреждать пользователей о тех ситуациях, где они могут по- вредить системе или данным; еще лучше, если у пользователя существует возможность отменить или исправить выполненные действия.

4. *Принцип «обратной связи».*

Всегда следует обеспечивать обратную связь для действий пользователя. Каждое действие пользователя должно получать визуальное, а иногда и звуковое подтверждение того, что программное обеспечение восприняло введенную команду; при этом вид реакции, по возможности, должен учитывать природу выполненного действия.

Обратная связь эффективна в том случае, если она реализуется своевременно, т.е. как можно ближе к точке последнего взаимодействия пользователя с системой. Когда компьютер обрабатывает поступившее задание, полезно предоставить пользователю информацию относительно текущего состояния процесса, а также возможность прервать этот процесс в случае необходимости.

5. *Простота*

Под простотой понимается легкость в изучении и использовании интерфейса. Кроме того, он должен предоставлять доступ ко всему перечню функциональных возможностей приложения. Один из возможных путей реализации простоты — представление на экране информации, минимально необходимой для выполнения очередного шага задания. Элементы должны размещаться на экране с учетов их смыслового значения и логической взаимосвязи.

6. *Гибкость*

Гибкость интерфейса — это его способность учитывать уровень подготовки и производительность труда пользователя. Свойство гибкости предполагает возможность изменения структуры диалога и/или входных данных.

7. *Эстетическая привлекательность.*

2.2 Графический интерфейс пользователя GUI

В основу разработки практически любого графического интерфейса пользователя (Graphical User Interface — GUI) положены три метафоры.

1. *Рабочий стол*. Для человека, сидящего за рабочим столом, доступны как определенные источники информации, так и средства обработки информации. При этом на рабочем столе могут находиться документы, представленные в различной форме: текст, таблицы, рисунки и т.д. В GUI рабочий стол воплощен пространством экрана монитора.

2. *Работаешь с тем, что видишь.* В каждый момент времени «сидящий» за рабочим столом может работать только с теми документами, которые он видит перед собой. Если необходимый документ отсутствует на столе, его необходимо достать (например, из папки). Объекты, необходимые для решения задачи, представлены в GUI в виде соответствующих графических образов (пиктограмм и окон).

3. *Видишь, что получил.* Выполняя какие-то действия над документами, человек тут же видит результат своей деятельности.

Все необходимые действия выполняются в GUI не с помощью команд, а с помощью *прямого манипулирования объектами*.

DCD — разработка, управляемая данными — означает, что проектирование интерфейса поддерживает такую модель взаимодействия пользователя с системой, при которой первичными являются обрабатываемые данные, а не требуемые для этого программные средства. При таком подходе внимание пользователя концентрируется на тех данных, с

которыми он работает, а не на поиске и загрузке необходимого приложения. При использовании DCD-технологии основным программным объектом является *документ* (некоторое абстрактное устройство хранения данных, используемых для выполнения заданий пользователей и их взаимодействия). DCD обуславливает необходимость применения объектно-ориентированного подхода к GUI. Т.е., объекты имеют характеристики, называемые свойствами (цвет, размер, дата модификации), допустимые операции (действия, которые могут быть выполнены над объектами, например перемещение или копирование), а также отношения между ними (набор, объединение, композиция, контейнер). Объекты, обладающие аналогичными свойствами и поведением, относятся к одному *классу*.

2.3 Концепция АРМ

АРМ — автоматизированное рабочее место системы управления, оборудованное средствами, обеспечивающими участие человека в реализации автоматизированных функций АСУ. Представляет собой персональный компьютер, оснащенный совокупностью профессионально ориентированных функциональных и обеспечивающих информационных технологий и размещенный на рабочем месте

АРМ содержит в себе функциональную информационную технологию — полностью или частично. Какая именно ее часть закрепляется за тем или иным АРМ, определяется декомпозицией целей в структуре управления объектом. Такое распределение функциональных информационных технологий на АРМ не должно нарушать требований самой предметной технологии. Наложение функциональных информационных технологий на управленческую структуру позволяет создать распределенную систему решения предметных задач.

АРМ включает три основных компонента: обучающую систему, комплекс программных продуктов по обработке информации и сервисные средства.

На первых этапах эволюции ИТ использовалась противоположная концепции АРМ *концепция вычислительного центра*, предполагавшая концентрацию основных вычислительных мощностей в одном отделе (ВЦ) и централизованную обработку данных. В отличие от ВЦ АРМ позволяет проводить обработку информации в местах ее непосредственного возникновения и использования. В результате все технологические процедуры, начиная от ввода информации и кончая получением выходных данных, могут выполняться работниками управления непосредственно на своих рабочих местах.

Выделяют следующие *основные признаки АРМ:*

—максимальная ориентация на конечного пользователя, достигаемая созданием инструментальных средств адаптации АРМ к уровню подготовки пользователя, возможностей его обучения и самообучения; размещение вычислительной техники непосредствен- но (или вблизи) на рабочем месте пользователя;

— ориентация АРМ на решение определенного класса задач, объединенных общей технологией обработки информации, единство режимов работы и эксплуатации;

— формализация профессиональных знаний, т.е. возможность предоставления с помощью АРМ самостоятельно автоматизировать новые функции и решать новые задачи в процессе накопления опыта работы с системой;

— модульность построения, обеспечивающая сопряжение АРМ с другими элементами системы обработки информации (системность), а также модификацию и наращивание возможностей АРМ без прерывания его функционирования (гибкость);

— эргономичность, т.е. создание для пользователя комфортных условий труда и дружественного, удобного интерфейса общения с системой.

Множество известных АРМ может быть классифицировано на основе следующих обобщенных критериев:

—функциональная сфера использования (научная деятельность, проектирование, производственно-технологические процессы, организационное управление);

—тип используемой ЭВМ (микро-, мини-, макро ЭВМ);

—режим эксплуатации (индивидуальный, групповой, сетевой);

—квалификация пользователей (профессиональные и непрофессиональные).

Внутри каждой из выделенных групп АРМ может быть проведена более детальная классификация.

3. СЕТЕВЫЕ ИТ

3.1 КОМПЬЮТЕРНЫЕ СЕТИ

Компьютерная сеть – система связи компьютеров и/или компью- терного оборудования. Для передачи информации могут быть использованы различные физические явления, как правило, различные виды электрических сигналов, световых сигналов или электромагнитного излучения.

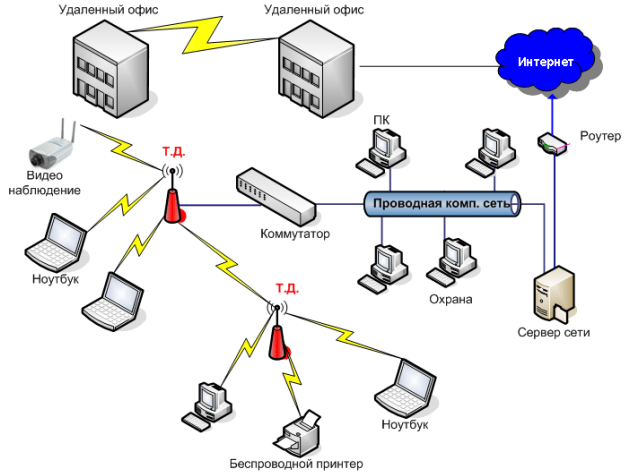


Рис. 3.1. Организация компьютерной сети

Локальные сети.

Локальные сети (LAN, Local Area Network) имеют замкнутую ин- фраструктуру до выхода на поставщиков услуг. Термин «LAN» может описывать и маленькую офисную сеть, и сеть уровня большого завода, занимающего несколько сотен гектаров. Локальные сети являются се- тями закрытого типа, доступ к ним разрешен только ограниченному кругу пользователей, для которых работа в такой сети непосредственно связана с их профессиональной деятельностью.

Наиболее точно было бы определить как локальную такую сеть, которая позволяет пользователям не замечать связи. Еще можно сказать, что локальная сеть должна обеспечивать прозрачную связь. По сути, компьютеры, связанные локальной сетью, объединяются в один вирту- альный компьютер, ресурсы которого могут быть доступны всем поль- зователям, причем этот доступ не менее удобен, чем к ресурсам, входя- щим непосредственно в каждый отдельный компьютер. Под удобством в данном случае понимается высокая реальная скорость доступа, ско- рость обмена информацией между приложениями, практически неза- метная для пользователя.

Топология локальных сетей – это физическое расположение ком- пьютеров сети друг относительно друга и способ соединения их линия- ми связи.

Шина (*bus*) – все компьютеры параллельно подключаются к одной линии связи. Информация от каждого компьютера одновременно пере- дается всем остальным компьютерам (рис. 3.2).

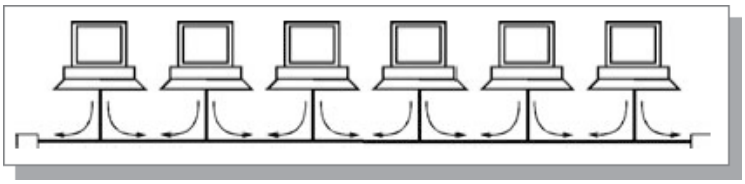


Рис. 3.2. Топология шина

Каждый компьютер должен иметь сетевую карту (рис. 6.3).

Соединение осуществляется коаксиальным кабелем (BNC). Скорость передачи данных – 10 Мб/с.

Звезда (*star*) – к одному центральному компьютеру или сетевому устройству присоединяются остальные периферийные компьютеры, причем каждый из них использует отдельную линию связи (рис. 3.3).

Каждый компьютер должен иметь интегрированную или отдель- ную сетевую карту.

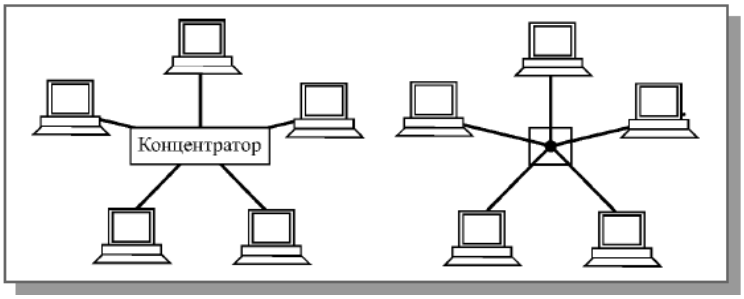


Рис. 3.3. Топология «звезда»

В качестве центрального устройства используют:

- концентратор (хаб, *hub*) – распространяет сигнал от одного подключенного устройства ко всем остальным ;
- коммутатор (свич, *switch*) – переключатель, передаёт данные непосредственно получателю, коммутирует каналы связи.

Кольцо (*ring*) – компьютеры по- следовательно объединены в кольцо (рис. 3.4). Передача информации в кольце всегда производится только в одном направлении. Каждый из ком- пьютеров передает информацию толь- ко одному компьютеру, следующему в цепочке за ним, а получает информа- цию только от предыдущего в цепочке компьютера.

Смешанные топологии. Чаще встречаются смешанные топологии: «дерево» – шина-звезда; звездно-кольцевая; сеточная топология.

Беспроводная сеть (wifi)

В последнее время получила широкое распространение технология беспроводных сетей WiFi. Ее можно отнести к топологии «звезда».

Каждый компьютер должен иметь сетевой адаптер внутренний или в виде отдельного USB-устройства.

В качестве центрального устройства используют точку доступа (рис. 6.12). Скорость передачи данных в сети WiFi – до 300 Мб/с, сильно зависит от числа подключенных компьютеров.

Состав локальной сети

Абонент (узел, хост, станция) – это устройство, подключенное к сети и активно участвующее в информационном обмене. Чаще всего абонентом (узлом) сети является компьютер, но абонентом также может быть, например, сетевой принтер или другое периферийное устройство, имеющее возможность напрямую подключаться к сети. Далее в курсе вместо термина «абонент» для простоты будет использоваться термин «компьютер».

Сервером называется абонент (узел) сети, который предоставляет свои ресурсы другим абонентам, но сам не использует их ресурсы. Таким образом, он обслуживает сеть. Серверов в сети может быть несколько, и совсем не обязательно, что сервер – самый мощный ком- пьютер. Выделенный (*dedicated*) сервер – это сервер, занимающийся только сетевыми задачами. Невыделенный сервер может, помимо обслуживания сети, выполнять и другие задачи. Специфический тип сервера – это сетевой принтер.

Клиентом называется абонент сети, который только использует се- тевые ресурсы, но сам свои ресурсы в сеть не отдает, т.е. сеть его об- служивает, а он ею только пользуется. Компьютер-клиент также часто называют рабочей станцией. В принципе каждый компьютер может быть одновременно как клиентом, так и сервером.

Под сервером и клиентом часто понимают также не сами компью- теры, а работающие на них программные приложения. В этом случае то приложение, которое только отдает ресурс в сеть, является сервером,а то приложение, которое только пользуется сетевыми ресурсами, – клиентом.

Маршрутизатор, роутер (*Router*) – сетевое устройство, на основании информации о топологии сети и определённых правил принимающее решения о пересылке пакетов данных между различными сегментами сети. Обычно маршрутизатор использует адрес получателя, указанный в пакетах данных, и определяет по таблице маршрутизации путь, по которому следует передать данные. В основном их применяют для объединения сетей разных типов, зачастую несовместимых по архитектуре и протоколам, например для объединения локальных сетей. Нередко маршрутизатор используется для обеспечения доступа из ло- кальной сети в глобальную сеть Интернет, осуществляя функции трансляции адресов и межсетевого экрана.

Приемопередатчик (трансивер) – служит для передачи информа- ции между адаптером и кабелем сети или между двумя сегментами (частями) сети. Трансиверы усиливают сигналы, преобразуют их уровни или преобразуют сигналы в другую форму (например, из электрической в световую и обратно). Трансиверами также часто называют встроенные в адаптер приемопередатчики.

Повторитель (*repeater*) – восстанавливает ослабленные сигналы (их амплитуду и форму), приводя их к исходному виду. Цель такой ретрансляции сигналов состоит исключительно в увеличении длины сети.

Мост (*bridge*) – сетевое устройство, предназначенное для объединения сегментов (подсети) компьютерной сети разных топологий и архитектур.

Шлюз (*gateway*) – аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной).

Межсетевой экран (*firewall, brandmauer*) – комплекс аппаратных или программных средств, осуществляющий контроль и фильтрацию проходящих через него сетевых пакетов в соответствии с заданными правилами.

Глобальные компьютерные сети

Глобальная сеть (WAN – Wide Area Network) – компьютерная сеть, охватывающая большие территории и включающая в себя большое чис- ло компьютеров.Глобальные сети отличаются от локальных тем, что рассчитаны на неограниченное число абонентов и используют, как правило, не слиш- ком качественные каналы связи и сравнительно низкую скорость пере- дачи, а механизм управления обменом у них в принципе не может быть гарантированно скорым.

Крупнейшими глобальными сетями являются:

Fidonet – международная любительская компьютерная сеть, по- строенная по технологии «из точки в точку». Особенностью Fidonet, определившей широкое распространение этой сети в России, является фактическая бесплатность подключения и использования ресурсов сети. Internet – всемирная система объединённых компьютерных сетей, построенная на использовании протокола IP и маршрутизации пакетов данных. Интернет образует глобальное информационное пространство, служит физической основой для Всемирной паутины и множества дру- гих систем (протоколов) передачи данных. Часто упоминается как

«Всемирная сеть» и «Глобальная сеть».

К сети Интернет подключаются городские сети, локальные сети и одиночные пользователи – абоненты (рис. 3.4). Подключение, как правило, происходит через провайдеров интернет-услуг.

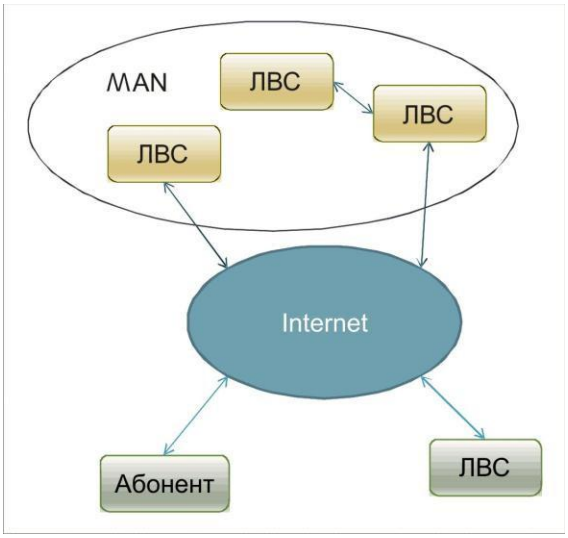


Рис. 3.4. Общая структура глобальной сети

Провайдер – организация, предоставляющая услуги доступа к Интернету и иные, связанные с Интернетом, услуги. Провайдер имеет скоростной канал связи с Интернетом и подключает к нему абонентов.

Операторы интернет-доступа по функционально-географическому признаку разделяются на следующие типы:

1. магистральные операторы (Internet Backbone Provider, IBP) – операторы межрегиональных IP-сетей, обеспечивающие перенос интернет-трафика между регионами страны и за рубеж;

2. конечные провайдеры (интернет-сервис-провайдеры, ISP) – обеспечивающие предоставление интернет-услуг конечным пользователям;

- региональные ISP – действующие на региональном уровне;
- локальные ISP – действующие на локальном уровне.

Операторы контент-ресурсов – компании, деятельность которых направлена на информационное наполнение сети Интернет и предоставление конечным пользователям услуг информационного характера. Различают следующие типы операторов контент-ресурсов:

3. операторы дата-центров, предоставляющие возможность своим пользователям разместить их информацию на серверах и сделать ее доступной в сети Интернет;

4. провайдеры приложений (ASP – Application Service Provider) – компании, предоставляющие пользователям программные приложения, использующие интернет-технологии и доступные из сети Интернет;

5. контент-провайдеры (CSP – Content Service Provider) – компании, предоставляющие пользователям Интернета ту или иную информацию или службу (контент).

Способы подключения к провайдеру

Подключение к провайдеру может осуществляться разными способами:

- Обычное модемное соединение. Скорость передачи данных – до 56 Кб/с:
- коммутируемое (Dial Up). Модем использует телефонную сеть общего пользования для подключения к другому компьютеру (серверу доступа). Обычно dial-up называют только доступ в Интернет на домашнем компьютере или удаленный модемный доступ в корпоратив- ную сеть. При этом во время сеанса связи телефонная линия абонента занята;
- по выделенной линии. Модем использует выделенную телефонную линию для подключения к серверу доступа. Такое соединение используется чаще всего организациями из-за значительных затрат на выделение отдельной линии.
- ISDN-модем. Скорость передачи данных – 128 Кб/с. Коммутируемый способ подключения к сети, обеспечивающий цифровые соединения между оконечными устройствами по телефонной сети общего пользования для поддержки речевых и информационных услуг, доступ к которым осуществляется с помощью набора стандартизированных интерфейсов. Позволяет совместить услуги телефонной связи и обмена данными.
- ADSL-модем. Постоянный способ подключения к Интернету, в котором доступная полоса пропускания канала распределена между исходящим и входящим трафиком асимметрично. Так как у большинства пользователей объем входящего трафика значительно превышает объем исходящего, то скорость исходящего трафика значительно ниже.Скорость приема данных – 8 Мб/с, скорость передачи данных – 1 Мб/с. ADSL позволяет работать в Интернете и говорить по телефону одно- временно.
- Ethernet-соединение – это способ постоянного подключения к Интернету по отдельному кабелю TP (витая пара). Применение такого доступа позволяет потребителям подключаться к сети со скоростью 100 Мбит/с или 1 Гбит/с при высоком качестве соединения.

- GPRS-соединение. Надстройка над технологией мобильной связи GSM, осуществляющая пакетную передачу данных. GPRS позволяет пользователю сети сотовой связи производить обмен данными с другими устройствами в сети GSM и с внешними сетями, в том числе Интернет.
- 3G-соединение. Технологии мобильной связи 3 поколения – набор услуг, который объединяет как высокоскоростной мобильный доступ с услугами сети Интернет, так и технологию радиосвязи, которая создаёт канал передачи данных.
- Radio Ethernet. Беспроводная связь, или связь по радиоканалу, сегодня используется и для построения магистралей (радиорелейные линии), и для создания локальных сетей, и для подключения удаленных абонентов к сетям и магистралям разного типа. Обеспечивает пропускную способность до 54 Мбит/с и позволяет создавать защищенные беспроводные каналы для передачи мультимедийной информации.
- Wi-Fi-соединение. Беспроводное соединение, требует непосредственной близости расположения точки доступа провайдера.
- Инфракрасный порт (IrDA, ИК-порт) – передача данных с ис- пользованием инфракрасного диапазона световых волн в качестве носителя. Является разновидностью атмосферной оптической линии связи ближнего радиуса действия.
- Спутниковые каналы связи. Используют для доступа в Интернет сами провайдеры или крупные предприятия.
- Оптоволоконные каналы связи. Обычно используются для соединения сегментов сети или для подключения к Интернету крупных абонентов.

4. БАЗОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Базовые информационные технологии – это кирпичики, из которых строятся прикладные информационные технологии. Пакеты прикладных программ используют и реализуют необходимые совокупности базовых ИТ.

4.1 Телекоммуникационные технологии

Телекоммуникационные технологии подразумевают передачу информации с помощью сигналов по электрическим проводам, волоконно-оптическому кабелю или радиоволнам. Передаваемые сигналы могут быть аналоговыми и цифровыми. В последнее время наблюдается переход на цифровую технологию, независимо от типа передаваемой ин- формации (звук, видео, текст и др.).

Мини-АТС – традиционно применяемое решение для организации голосовой связи в масштабах предприятия.

Сотовая связь. С появлением корпоративных сотовых тарифов оказалось дешевле и удобнее организовать сотовую связь внутри пред- приятия.

Компьютерная сеть. Сетевые технологии на сегодняшний момент используются на большинстве предприятий.

Комплексные решения. С появлением IP-телефонии наконец-то удалось при помощи компьютерной сети эффективно объединить функции голосовой и компьютерной связи.

4.2 Технологии создания и обработки текста

Традиционно создание текстовой информации производится при помощи его набора на клавиатуре. Однако возможно получить текст, отсканировав его на сканере. При этом используется технология распознавания текста (OCR).

Оптическое распознавание символов (optical character recognition, OCR) – это механический или электронный перевод изоб- ражений рукописного, машинописного или печатного текста в последовательность кодов, использующихся для представления в текстовом редакторе. Распознавание широко используется для конвертации книг и документов в электронный вид, для автоматизации систем учета в бизнесе или для публикации текста на Web-странице.

Распознавание речи – процесс преобразования речевого сигнала в текстовый поток; развивающаяся технология создания текстовой информации.

Обработка текста (редактирование, форматирование, подготовка к публикации) осуществляется текстовым редактором или текстовым процессором.

Гипертекстовая технология. Гипертекст представляется набором текстов, содержащих узлы перехода (гиперссылки) между ними, которые позволяют избирать читаемые сведения или последовательность чтения. Общеизвестным и ярко выраженным примером гипертекста служат Web-страницы – документы HTML (язык разметки гипертекста), размещённые в Сети. В более широком понимании термина гипертекстом является любая повесть, словарь или энциклопедия, где встречаются отсылки к другим частям данного текста, имеющие отношения к данному термину. В компьютерной терминологии гипертекст – текст, сформированный с помощью языка разметки, потенциально содержащий в себе гиперссылки. Большинство текстовых процессоров поддер- живают гипертекстовую технологию.

4.3 Технологии электронных таблиц

Электронная таблица – компьютерная программа, позволяющая проводить вычисления с данными, представленными в виде двухмер- ных массивов, имитирующих бумажные таблицы.

Электронные таблицы (ЭТ) представляют собой удобный инструмент для автоматизации вычислений. Многие расчёты, в частности в области бухгалтерского учёта, выполняются в табличной форме: ба- лансы, расчётные ведомости, сметы расходов и т.п. Кроме того, реше- ние численными методами целого ряда математических задач удобно выполнять именно в табличной форме. Использование математических формул в ЭТ позволяет представить взаимосвязь между различными параметрами некоторой реальной системы. Решение многих вычислительных задач, которые раньше можно было осуществить только с помощью программирования, стало возможно реализовать через математическое моделирование в электронной таблице.

5. ТЕХНОЛОГИИ СОЗДАНИЯ И ОБРАБОТКИ ГРАФИКИ

Растровая графика создается с помощью сканирования готового изображения, цифрового фотоаппарата или видеокамеры и с помощью захвата изображения с экрана (скриншот). В редких случаях пользуются растровым графическим редактором, т.к. гораздо удобнее нарисовать изображение в векторном редакторе и экспортировать (преобразовать) его в растровый формат.

Редактируют растровую графику в растровых графических редак- торах. Наиболее часто употребляемые операции редактирования:

- удаление эффекта красных глаз;
- удаление шумов (мелких дефектов);
- настройка яркости/контрастности/интенсивности;
- повышение резкости;
- коррекция цветов;
- поворот изображения;
- обрезка изображения, изменение разрешения изображения;
- применение художественных эффектов.

Инструменты рисования (например, работа кистью) требуют определенной подготовки пользователя. При работе с инструментами рисования можно получать комбинированные изображения, когда вновь создаваемые элементы не объединяются с фоном, а хранятся в виде объ- ектов, которые можно перемещать и трансформировать. Такие комбинированные рисунки можно сохранять только в индивидуальных форматах используемого редактора, при экспорте изображения в стандарт- ные растровые форматы объекты будут объединены с фоном.

6. Мультимедиатехнологии

Мультимедиатехнологии – комплекс аппаратных и программных средств, позволяющих пользователю работать в диалоговом режиме с разнородными данными (графика, текст, звук, видео), организованны- ми в виде единой информационной среды.

Термин «мультимедиа» зачастую используется для обозначения носителей информации, позволяющих хранить значительные объемы данных и обеспечивать достаточно быстрый доступ к ним (CD, DVD, HD). В таком случае термин «мультимедиа» означает, что компьютер может использовать такие носители и предоставлять информацию поль- зователю через все возможные виды данных, такие как аудио, видео, анимация, изображение и другие, в дополнение к традиционным способам предоставления информации, таким как текст.

Из программных средств к мультимедийным относят:

- видео- и аудиопроигрыватели;
- средства создания и редактирования видео- и аудиофайлов;
- программы записи на CD-, DVD-, HD-носители.

7.Технологии управления базами данных

База данных – совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации.

Классификация БД по модели данных:

• иерархические, могут быть представлены как дерево, состоящее из объектов различных уровней;

• сетевые, подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию;

• реляционные – основной тип современных баз данных, состоящие из таблиц, между которыми могут существовать связи по ключевым значениям;

• объектно-ориентированные, в которых данные оформлены в виде моделей объектов, включающих прикладные программы, которые управляются внешними событиями.

Система управления базами данных (СУБД) – комплекс программных и языковых средств, необходимых для создания и модификации базы данных, добавления, модификации, удаления, поиска и отбора информации, представления информации на экране и в печатном виде,разграничения прав доступа к информации, выполнения других операций с базой.

7.1 Реляционная БД

Таблица базы данных (*table*) – регулярная структура, которая состоит из однотипных строк (записей – *records*), разбитых на столбцы (поля – *fields*).

Ключевой элемент таблицы (ключ – regular key) – такое ее поле (простой ключ) или строковое выражение, образованное из значений нескольких полей (составной ключ), по которому можно определить значения других полей для одной или нескольких записей таблицы. На практике для использования ключей создаются индексы – служебная информация, содержащая упорядоченные сведения о ключевых значениях.

Первичный ключ (*primary key*) – главный ключевой элемент, однозначно идентифицирующий строку в таблице. Могут также суще- ствовать альтернативный (*candidate key*) и уникальный (*unique key*) ключи, служащие также для идентификации строк в таблице.

Связь (*relation*) – функциональная зависимость между объектами. В реляционных базах данных между таблицами устанавливаются связи по ключам, один из которых в главной (родительской – parent) таблице первичный, второй – внешний ключ – во внешней (дочерней – child) таблице, как правило, первичным не является и образует связь «один ко многим» (1:N). В случае первичного внешнего ключа связь между таблицами имеет тип «один к одному» (1:1). Информация о связях сохраняется в базе данных.

Внешний ключ (*foreign key*) – ключевой элемент подчиненной (внешней, дочерней) таблицы, значение которого совпадает со значением первичного ключа главной (родительской) таблицы.

Ссылочная целостность данных (*referential integrity*) – набор правил, обеспечивающих соответствие ключевых значений в связанных таблицах.

Репликация базы данных – создание копий базы данных (реплик), которые могут обмениваться обновляемыми данными или реплицированными формами, отчетами или другими объектами в результате выполнения процесса синхронизации.

7.2 Клиент-серверная технология

Клиент-сервер (*Client-server*) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемыми серверами, и заказчика- ми услуг, называемыми клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением

Преимущества клиент-серверной технологии:

• Делает возможным, в большинстве случаев, распределить функции вычислительной системы между несколькими независимыми компьютерами в сети. Это позволяет упростить обслуживание вычислительной системы. В частности, замена, ремонт, модернизация или перемещение сервера не затрагивают клиентов.

• Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиен- там с соответствующими правами доступа.

• Позволяет объединить различные клиенты. Использовать ресурсы одного сервера часто могут клиенты с разными аппаратными платформами, операционными системами и т.п.

Клиент-серверная СУБД – СУБД, использующая технологию «клиент-сервер».

Клиент-серверная СУБД позволяет обмениваться клиенту и серверу минимально необходимыми объёмами информации. При этом основ- ная вычислительная нагрузка ложится на сервер. Клиент может выполнять функции предварительной обработки перед передачей информации серверу, но в основном его функции заключаются в организации досту- па пользователя к серверу.

В большинстве случаев клиент-серверная СУБД гораздо менее требовательна к пропускной способности компьютерной сети, чем файл- серверная СУБД, особенно при выполнении операции поиска в базе данных по заданным пользователем параметрам, т.к. для поиска нет необходимости пересылать на клиентский компьютер весь массив данных: клиент передаёт параметры запроса серверу, а сервер производит поиск по полученному запросу в локальной базе данных. Результат вы

полнения запроса, который обычно на несколько порядков меньше по объёму, чем весь массив данных, возвращается клиенту, который обес- печивает отображение результата пользователю .

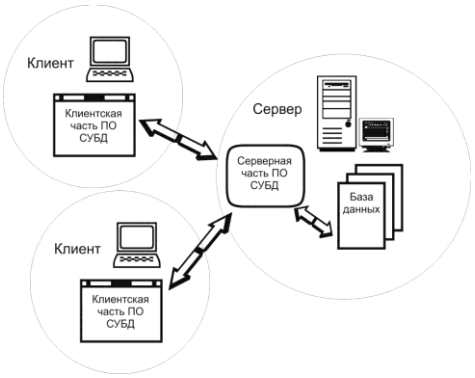


Рис. 7.5. Иллюстрация клиент-серверной технологии СУБД

8. ГЕОИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Геоинформационные технологии Геоинформационная система (ГИС – GIS) – информационная система, предназначенная для сбора, хранения, анализа и графической визуализации пространственных данных и связанной с ними информации о представленных в ГИС объектах. Термин также используется в более узком смысле – ГИС как инструмент (программный продукт), позволя- ющий пользователям искать, анализировать и редактировать цифровые карты, а также дополнительную информацию об объектах, например: высоту здания, адрес, количество жильцов.

ГИС включает в себя возможности систем управления базами данных (СУБД), редакторов растровой и векторной графики и аналитических средств и применяется в картографии, геологии, метеорологии, землеустройстве (кадастры), экологии, муниципальном управлении (проектирование коммуникаций), транспорте (навигация), экономике, строительстве (ландшафтное проектирование), обороне и многих других областях.

По территориальному охвату различают глобальные ГИС (*global GIS*), субконтинентальные ГИС, национальные ГИС, зачастую имеющие статус государственных, региональные ГИС (*regional GIS*), субрегио- нальные ГИС и локальные, или местные ГИС (*local GIS*).

ГИС различаются предметной областью информационного моделирования, например: городские ГИС, или муниципальные ГИС, МГИС (*urban GIS*), природоохранные ГИС (*environmental GIS*) и т.п.; среди них особое наименование (как особо широко распространённые) получили земельные информационные системы. Проблемная ориентация ГИС определяется решаемыми в ней задачами (научными и прикладными), среди них инвентаризация ресурсов (в том числе кадастр), анализ, оценка, мониторинг, управление и планирование, поддержка принятия решений. Интегрированные ГИС, ИГИС (*integrated GIS, IGIS*) совмещают функциональные возможности ГИС и систем цифровой обработки изображений (данных дистанционного зондирования) в единой инте-грированной среде.

Полимасштабные, или масштабно-независимые ГИС (*multiscale GIS*), основаны на множественных, или полимасштабных представлениях пространственных объектов (*multiple representation, multiscale representation*), обеспечивая графическое или картографическое воспроизведение данных на любом из избранных уровней масштабного ряда на основе единственного набора данных с наибольшим пространственным разрешением. Пространственно-временные ГИС (*spatio-temporal GIS*) оперируют пространственно-временными данными. Научные, технические, технологические и прикладные аспекты проектирования, создания и использования ГИС изучаются геоинформатикой.

9. ТЕХНОЛОГИИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Технологии искусственного интеллекта Искусственный интеллект (ИИ, AI – Artificial intelligence) – наука и технология создания интеллектуальных машин, особенно интеллекту-альных компьютерных программ. ИИ связан со сходной задачей ис- пользования компьютеров для понимания человеческого интеллекта, но не обязательно ограничивается биологически правдоподобными методами.

Это научное направление, в рамках которого ставятся и решаются задачи аппаратного или программного моделирования тех видов человеческой деятельности, которые традиционно считаются интеллектуальными.

Наука под названием «Искусственный интеллект» входит в ком- плекс компьютерных наук, а создаваемые на её основе технологии относятся к информационным технологиям. Задачей этой науки является воссоздание с помощью вычислительных систем и иных искусственных устройств разумных рассуждений и действий:

Моделирование рассуждений подразумевает создание символьных систем, на входе которых поставлена некая задача, а на выходе требуется её решение. Как правило, предлагаемая задача уже формализована, т.е. переведена в математическую форму, но либо не имеет алгоритма решения, либо он слишком сложен, трудоёмок и т.п. В это направление входят: доказательство теорем, принятие решений и теория игр, планирование и диспетчеризация, прогнозирование.

Обработка естественного языка, в рамках которого проводится анализ возможностей понимания, обработки и генерации текстов на «человеческом» языке. В частности, здесь ещё не решена проблема машинного перевода текстов с одного языка на другой.

Инженерия знаний, объединяющая задачи получения знаний из простой информации, их систематизации и использования. Достижения в этой области затрагивают почти все остальные направления исследований ИИ:

- машинное обучение – касается процесса самостоятельного получения знаний интеллектуальной системой в процессе её работы;
- экспертные системы – программы, использующие специализированные базы знаний для получения достоверных заключений по какой-либо проблеме;
- распознавание образов, например: распознавание символов, рукописного текста, речи, анализ текстов. Особо стоит упомянуть компьютерное зрение, которое связано ещё и с робототехникой.

10. WWW-технологии (World Web Wide)

Сеть WWW образуют миллионы Web-серверов, расположенных по всему миру. Web-сервер является программой, запускаемой на подклю- ченном к сети компьютере и передающей данные по протоколу HTTP.

Для определения местонахождения ресурсов в сети используются локаторы ресурсов URL (Uniform Resource Locator). Доменное имя (или IP-адрес) входит в состав URL для обозначения компьютера (его сетево- го интерфейса), на котором работает программа Web-сервер. Пример URL: http://portal.tpu.ru/student/life/clubs

На клиентском компьютере для просмотра информации, полученной от Web-сервера, применяется специальная программа – Web- браузер. Основная функция Web-браузера – отображение гипертекстовых страниц (Web-страниц). Для создания гипертекстовых страниц в

WWW изначально использовался язык HTML. Множество Web-страниц образуют Web-сайт.

Базовым протоколом сети гипертекстовых Web-ресурсов является протокол HTTP. В его основу положено взаимодействие «клиент- сервер», т.е. предполагается, что:

1. потребитель-клиент, инициировав соединение с поставщиком-сервером, посылает ему запрос;

2. поставщик-сервер, получив запрос, производит необходимые действия и возвращает обратно клиенту ответ с результатом.

При этом возможны два способа организации работы компьютера-клиента:

• тонкий клиент – это компьютер-клиент, который переносит все задачи по обработке информации на сервер. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с Web-приложениями;

• толстый клиент, напротив, производит обработку информации независимо от сервера, использует последний в основном лишь для хранения данных.

Сайт (*website*) – совокупность электронных документов (файлов) частного лица или организации в компьютерной сети, объединённая под одним адресом (доменным именем или IP-адресом). По умолчанию подразумевается, что сайт располагается в сети Интернет.

Страницы сайтов – это файлы с текстом, размеченным на языке HTML. Эти файлы, будучи загруженными посетителем на его компью- тер, обрабатываются браузером и выводятся на его средство отображения (монитор, экран КПК, принтер или синтезатор речи). Язык HTML позволяет форматировать текст, различать в нём функциональные элементы, создавать гипертекстовые ссылки (гиперссылки) и вставлять в отображаемую страницу изображения, звукозаписи и другие мультимедийные элементы. Отображение страницы можно изменить добавлением в неё таблицы стилей на языке CSS или сценариев на языке JavaScript.

Страницы сайтов могут быть простым статичным набором файлов или создаваться специальной компьютерной программой на сервере – так называемым движком сайта. Движок может быть либо сделан на заказ для отдельного сайта, либо быть готовым продуктом, рассчитанным на некоторый класс сайтов. Некоторые из движков могут обеспечить владельцу сайта возможность гибкой настройки структурирования и вывода информации на Web-сайте. Такие движки называются системами управления содержимым (CMS). Язык HTML позволяет размечать электронный документ, который отображается на экране с полиграфическим уровнем оформления; результирующий документ может содержать самые разнообразные метки, иллюстрации, аудио- и видеофрагменты и т.д. В состав языка вошли развитые средства для создания различных уровней заголовков, шрифтовых выделений, различные списки, таблицы и многое другое.

Таким образом, гипертекстовая база данных в концепции WWW – это набор текстовых файлов, размеченных на языке HTML, который определяет форму представления информации (разметка) и структуру связей между этими файлами и другими информационными ресурсами (гипертекстовые ссылки). Гипертекстовые ссылки, устанавливающие связи между текстовыми документами, постепенно стали объединять самые различные информационные ресурсы, в том числе звук и видео; в результате возникло новое понятие – гипермедиа.

По доступности сервисов сайты бывают:

- открытые – все сервисы полностью доступны для любых посетителей и пользователей;
- полуоткрытые – для доступа необходимо зарегистрироваться (обычно бесплатно);
- закрытые – полностью закрытые служебные сайты организаций (в том числе корпоративные сайты), личные сайты частных лиц. Такие сайты доступны для узкого круга людей. Доступ новым людям обычно даётся через т.н. инвайты (приглашения).

По природе содержимого:

- статические – всё содержимое заранее подготавливается. Пользователю выдаются файлы в том виде, в котором они хранятся на сервере;
- динамические – содержимое генерируется специальными скриптами (программами) на основе других данных из любого источника. Сервер получает запрос от клиента и запускает обработку файла-скрипта интерпретатором. Языки программирования на серверной стороне используются разные, наиболее часто встречаются, например: PHP, Perl, ASP.NET, Ruby и др. Именно на этой стадии происходит вы- борка необходимой информации из баз данных и наполнение ею страницы, после чего она передаётся клиенту в ответ.

После того как страница получена клиентом с сервера, программа-браузер обрабатывает её и отображает пользователю, при этом исполняя скрипты клиентской стороны, если они были указаны в странице и получены. На клиентской стороне используется скрипт-программа, которая может применяться как для минимальных (невидимых глазу) операций (например, проверки на правильность задания паролей при регистрации на сайтах), так и глобальных последовательностей и приложений.

По физическому расположению:

- внешние сайты сети Интернет;
- локальные сайты – доступны только в пределах локальной сети. Это могут быть как корпоративные сайты организаций, так и сайты частных лиц в локальной сети провайдера.

По схеме представления информации, её объёму и категории решаемых задач можно выделить следующие типы Web-ресурсов:

- персональный сайт (физического лица) обычно содержит личную информацию, часто используется в социальных сетях, иногда применяется специалистами для размещщения своего резюме;
- интернет-представительства владельцев бизнеса (торговля и услуги, не всегда связанные напрямую с Интернетом):
- сайт-визитка – содержит самые общие данные о владельце сайта (организации или индивидуального предпринимателя): вид деятельности, история, прайс-лист, контактные данные, реквизиты, схема проезда и т.д.;
- представительский сайт – так иногда называют сайт-визитку с расширенной функциональностью: подробное описание услуг, портфолио, отзывы, форма обратной связи и т.д.;
- корпоративный сайт – содержит полную информацию о компании- владельце, услугах/продукции, событиях в жизни компании. Отличается от сайта-визитки и

представительского сайта полнотой представленной информации, зачастую содержит различные функциональные инструменты для работы с контентом (поиск и фильтры, календари событий, фотогалереи, корпоративные блоги, форумы). Может быть интегрирован с внутренними информационными системами компании-владельца (КИС, CRM, бухгалтерскими системами). Может содержать закрытые разделы для тех или иных групп пользователей – сотрудников, дилеров, контрагентов и пр.;

- каталог продукции – в каталоге присутствует подробное описание товаров/услуг, сертификаты, технические и потребительские данные, отзывы экспертов и т.д. На таких сайтах размещается информация о товарах/услугах, которую невозможно поместить в прайс-лист; интернет-магазин – сайт с каталогом продукции, с помощью которого клиент может заказать нужные ему товары. Используются различные системы расчётов: от пересылки товаров наложенным платежом или автоматической пересылки счета по факсу до расчётов с помощью пластиковых карт;
- промо-сайт – сайт о конкретной торговой марке или продукте, на таких сайтах размещается исчерпывающая информация о бренде, различных рекламных акциях (конкурсы, викторины, игры и т.п.);
- информационные ресурсы:
- тематический сайт – сайт, предоставляющий специфическую узкотематическую информацию о какой-либо теме;
- тематический портал – это очень большой Web-ресурс, который предоставляет исчерпывающую информацию по определённой тематике. Порталы похожи на тематические сайты, но дополнительно содержат средства взаимодействия с пользователями и позволяют пользователям общаться в рамках портала (форумы, чаты) – это среда существования пользователя;
- Web-сервис – услуга, созданная для выполнения каких-либо задач в рамках сети WWW:
- доски объявлений;
- каталоги сайтов;
- поисковые сервисы – например: Yahoo!, Google;
- почтовые сервисы (с использованием Web-интерфейса);
- Web-форумы;
- блоговые сервисы;
- файлообменный (пиринговый) сервис, например Bittorrent;
- датахостинговый (хранение данных) сервис, например Skydrive;
- датаэдиторинговый (редактирование данных) сервис, например Google Docs;
- Фотохостинг, например: Picnik, ImageShack, Panoramio, Photobucket;
- хранение видео, например: YouTube, Dailymotion;
- социальные медиа, например Buzz;
- комбинированные Web-сервисы (социальные сети), например: Facebook, Twitter;
- комбинированные Web-сервисы (специализированные социальные сети), например: MySpace, Flickr; конструкторы сайтов позволяют быстро создать сайт на своей платформе с использованием большого числа заготовок и шаблонов. (Например: uCoz, Jimdo, Narod.ru).
- Хостинг – размещение сайта на компьютере. После разработки сайта необходимо его разместить на Web-сервере провайдера или собственном. После приобретения доменного имени для своего сайта необходимо связать имя сайта с IP-адресом Web-сервера, на котором он размещен путем добавления специальных строк-описаний, хранящихся на DNS-серверах. Затем, когда ваш сайт станет доступен в Интернете по его доменному имени, можно приступать к раскрутке сайта.

Электронная почта

Электронная почта (e-mail) – технология и предоставляемые ею услуги по пересылке и получению электронных сообщений (называемых «письма», или «электронные письма») по распределённой (в том числе глобальной) компьютерной сети.

Основным отличием (и достоинством е-майл) от прочих систем передачи сообщений (например, служб мгновенных сообщений) ранее являлась возможность отложенной доставки сообщения, а также развитая (и запутанная из-за длительного времени развития) система взаимодействия между независимыми почтовыми серверами (отказ одного сервера не приводил к неработоспособности всей системы).

В настоящее время любой начинающий пользователь может завести свой бесплатный электронный почтовый ящик, достаточно зарегистрироваться на одном из интернет-порталов.

Функционирование электронной почты невозможно без почтовых серверов. Это клиент-серверная технология. На компьютере пользователя установлена клиентская часть – обычно почтовый клиент или просто Web-браузер. На сервере работает программа – почтовый сервер. Почтовый сервер может физически находиться в локальной сети, где находится компьютер пользователя, у провайдера интернет-доступа или где-нибудь в Интернете. Чем ближе он расположен, тем надежнее и быстрее будет доступ к почте, хотя это и не всегда так.

Общение конечного пользователя с почтовой системой может быть принципиально двух типов:

• при помощи обычного Web-браузера. Для этого необходимо, чтобы почтовый сервер предоставлял пользователям Web-интерфейс. В этом случае вся база сообщений хранится на сервере. Пользователь может получить доступ ко всей своей почте с любого компьютера. В ряде случаев это удобно и очень мобильно, однако исключает обработку полученной почты в Off-Line (без доступа к Интернет). Большинство пользователей бесплатных почтовых сервисов (mail.ru, gmail.com, yandex.ru и др.) используют именно такой способ;

• при помощи почтового клиента, установленного на своем компьютере (например, Outlook Express). В этом случае вся база сообщений хранится на компьютере пользователя. В момент запуска почтового клиента весь почтовый ящик пользователя (сообщения, накопленные с момента последней выгрузки) выгружается на его компьютер. Пользователь может получить доступ к своей почте только со своего компьютера. В последнее время многие почтовые службы стали предоставлять пользователям возможность доступа к почте по Web-интерфейсу. Это сделало возможным прочитать последние (ещё не выгруженные) сообщения при помощи обычного Web-браузера с любого другого компьютера.

Независимо от того, каким способом пользователь отправляет письмо, его сообщение попадает в очередь отправки почтового сервера.

Для своих пользователей серверы почтовой системы являются релеями (пользователи отправляют почту не на серверы почтовой системы адресата, а на «свой» почтовый сервер, который передаёт письма далее). Общепринятым в мире протоколом обмена электронной почтой является SMTP (Simple mail transfer protocol – простой протокол передачи почты). В общепринятой реализации он использует DNS для определения правил пересылки почты.

После попадания почты на конечный сервер он осуществляет временное или постоянное хранение принятой почты. Существует две различные модели работы с почтой: концепция почтового ящика и хранилища почты. В концепции почтового ящика почта на сервере хранится временно, в ограниченном объёме (аналогично почтовому ящику для бумажной почты), а пользователь периодически обращается к ящику и «забирает»

письма (т.е. почтовый клиент скачивает копию письма к себе и удаляет оригинал из почтового ящика). На основании этой концепции действует протокол POP3.

Концепция постоянного хранения подразумевает, что вся корреспонденция, связанная с почтовым ящиком (включая копии отправленных писем), хранится на сервере, а пользователь обращается к хранилищу (иногда его по традиции также называют «почтовым ящиком») для просмотра корреспонденции (как новой, так и архива) и написания новых писем (включая ответы на другие письма). На этом принципе действует протокол IMAP и большинство Web-интерфейсов бесплатных почтовых служб. Подобное хранение почтовой переписки требует значительно бóльших мощностей от почтовых серверов, в результате во многих случаях происходит разделение между почтовыми серверами, пересылающими почту, и серверами хранения писем.

Почтовые рассылки. Почтовая система позволяет организовать сложные системы, основанные на пересылке почты от одного ко многим абонентам. Это:

• почтовые рассылки – письмо от одного адреса с одинаковым (или меняющимся по шаблону) содержимым, рассылаемое подписчикам рассылки. Технически может быть организовано как отправка множе- ства писем (используется при шаблонных письмах) или как отправка письма с множеством получателей (в полях TO, CC, BCC). Для управ- ления крупными почтовыми рассылками (более 10–50 абонентов) ис- пользуются специализированные программы (например, mailman). Пра- вильно организованная почтовая рассылка должна контролировать возврат писем (сообщения о невозможности доставить письмо) с исключением недоступных адресатов из списка рассылки, позволять подписчикам отписываться от рассылок. Нежелательные почтовые рассылки называются спамом и существенно осложняют функционирование почтовых систем;

• группы переписки – специализированный тип почтовой рассылки, в которой письмо на адрес группы (обычный почтовый адрес, обработкой почты которого занимается специализированная программа) рассылается всем участникам группы. Является аналогом новостных конференций, эхоконференций. Правильно настроенная почтовая рассылка должна контролировать циклы (два робота рассылок, подписанные друг на друга, способны создать бесконечный цикл пересылки писем), ограничивать список участников рассылки, имеющих право на помещение сообщения, выполнять прочие требования к почтовой рассылке.

Для управления почтовыми рассылками используются менеджеры почтовых рассылок. Помимо ведения списка адресов и выполнения отсылки заданного сообщения они обеспечивают фильтрацию писем, возможности премодерации писем перед помещением в рассылку, ведение архивов, управление подпиской/отпиской, рассылку дайджестов (краткого содержимого) вместо всего объёма рассылки.

Спам – разновидность почтовой рассылки с целью рекламы (часто нежелательной) того или иного товара или услуги, аналог бумажной рекламы, бесплатно распространяемой по почтовым ящикам жилых домов.

По мере ужесточения запрета на размещение рекламы сообщения разделились на легитимные рассылки (на которые обычно подписывается пользователь и от которых он может отказаться в любой момент) и нелегитимные (собственно, и называемые спамом). Для борьбы со спамом были разработаны различные механизмы (чёрные списки отправителей, серые списки, требующие повторного обращения почтового сервера для отправки, контекстные фильтры). Одним из последствий внедрения средств борьбы со спамом стала вероятность «ошибочно положительного» решения относительно спама, т.е. часть писем, не являющихся спамом, стала помечаться как спам. Агрессивная антиспам-политика (уничтожение писем, кажущихся спамом, в автоматическом режиме без

уведомления отправителя/получателя) приводит к трудно обнаруживаемым проблемам с прохождением почты.

Чат

Чат (*chat* – болтать) – средство обмена сообщениями по компьютерной сети в режиме реального времени, а также программное обеспечение, позволяющее организовывать такое общение. Характерной особенностью является коммуникация именно в реальном времени или близкая к этому, что отличает чат от форумов и других «медленных» средств.

Под словом «чат» обычно понимается групповое общение, хотя к ним можно отнести и обмен текстом «один на один» посредством программ мгновенного обмена сообщениями, например ICQ или даже SMS.

Существует несколько разновидностей программной реализации чатов:

- HTTP, или Web-чаты. Такой чат выглядит как обычная Web- страница, где можно прочесть последние несколько десятков фраз, написанные участниками чата и модераторами. Страница чата автоматически обновляется с заданной периодичностью;
- чаты, использующие технологию Adobe Flash. Вместо периодической перезагрузки страницы между клиентом и сервером открывается сокет, что позволяет моментально отправлять или получать сообщения, расходуя меньше трафика;
- IRC – специализированный протокол для чатов;
- программы-чаты для общения в локальных сетях (например: Vypress Chat, Intranet Chat, Pichat). Часто есть возможность передачи файлов;
- чаты, реализованные поверх сторонних протоколов (например чат, использующий ICQ);
- чаты, работающие по схеме клиент-сервер, это позволяет использовать их в сетях со сложной конфигурацией, а также управлять клиентскими приложениями (например: Mychat, Jabber).

Видеоконференции

Видеоконференция – это область информационной технологии, обеспечивающая одновременно двустороннюю передачу, обработку, преобразование и представление интерактивной информации на расстояние в реальном режиме времени с помощью аппаратно-программных средств вычислительной техники.

Программные решения устанавливаются на компьютер, оснащённый web-камерой и головной гарнитурой. К данному классу относятся:

- платные решения: Tandberg Movi, Meeting point 4.0, Polycom PVX, ВидеоПорт VCS, Tandberg See&Share и т.д.;
- бесплатные решения: Skype, ВидеоПорт Online, NetMeeting, VC software, Ekiga, PalTalk, Visitalk, Adobe Acrobat Connect Pro meetings и др.

Платные решения, в отличие от бесплатных, обычно обеспечивают более широкие функциональные возможности при проведении конференций (например, поддерживается большое число участников) и совместимость с аппаратными решениями видеоконференцсвязи различ- ных производителей благодаря использованию открытых стандартов.

Web-конференции – технологии и инструменты для онлайн-встреч и совместной работы в режиме реального времени через Интернет. Web-конференции позволяют проводить онлайн-презентации, совместно работать с документами и приложениями, синхронно просматривать сайты, видеофайлы и изображения. При этом каждый участник находится на своём рабочем месте за компьютером.

Web-конференции, которые предполагают «одностороннее» вещание спикера и минимальную обратную связь от аудитории, называют вебинарами.

Вебинар означает особый тип Web-конференций. Связь, как правило, односторонняя – со стороны говорящего, и взаимодействие со слушателями ограничено. Вебинары могут быть совместными и вклю- чать в себя сеансы голосований и опросов, что обеспечивает полное взаимодействие между аудиторией и ведущим. В некоторых случаях ведущий может говорить через телефон, комментируя информацию, отображаемую на экране, а слушатели могут ему отвечать (предпочтительно по телефону с громкоговорителем). На рынке также присутствуют технологии, в которых реализована поддержка VoIP. Это аудиотехнологии, обеспечивающие полноценную аудиосвязь через сеть. Вебинары (в зависимости от провайдера) могут обладать функцией анонимности или «невидимости» пользователей, благодаря чему участники одной и той же конференции могут не знать о присутствии друг друга.

11. ТЕХНОЛОГИИ СОЗДАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

11.1 Общая характеристика технологии создания программного обеспечения

К программно-инструментальным средствам в первую очередь относятся алгоритмические языки и соответствующие им трансляторы, затем системы управления базами данных (СУБД) с языковыми средствами программирования в их среде, электронные таблицы с соответствующими средствами их настройки и т.п.

Рассмотрим этапы разработки программ.

Рис.11.1 - Этапы разработки программного обеспечения.

Первый этап представляет собой *постановку задачи.* На этом этапе раскрывается сущность задачи, т.е. формулируется цель ее решения; определяется взаимосвязь с другими задачами; указывается периодичность решения; устанавливаются состав и формы представления входной, промежуточной и результатной информации.

Особое внимание в процессе постановки задачи уделяется детальному описанию входной, выходной (результатной) и межуточной информации.

Особенность реализации этого этапа технологического процесса заключается в том, что конечный пользователь разрабатываемой программы, хорошо знающий ее проблемную сторону, обычно хуже представляет специфику и возможности использования ЭВМ для решения задачи. В свою очередь, предметная область пользователя (особенно ее отдельные нюансы, способные оказать влияние на решение задачи) зачастую незнакома разработчику программы, хотя он знает возможности и ограничения на применение ЭВМ. Именно эти противоречия являются основной причиной возникновения ошибок при реализации данного этапа технологического процесса разработки программ, которые затем неизбежно отражаются и на последующих этапах. Отсюда вся важность и ответственность этого этапа, требующего осуществления корректной и полной постановки задачи, а также необходимости однозначного ее понимания как разработчиком программы, так и ее пользователем.

Второй этап в технологии разработки программ - *математическое описание задачи* и выбор метода ее решения. Наличие этого этапа обусловливается рядом причин, одна из которых вытекает из свойства неоднозначности естественного языка, на котором описывается постановка задачи. В связи с этим на нем выполняется формализованное описание задачи, т.е. устанавливаются и формулируются средствами языка математики логико-математические зависимости между исходными и результатными данными. Математическое описание задачи обеспечивает ее однозначное понимание пользователем и разработчиком программы.

Сложность и ответственность этапа математического описания задачи и выбора (разработки) соответствующего метода ее решения часто требуют привлечения квалифицированных специалистов области прикладной математики, обладающих знанием таких дисцип-

лин, как исследование операций, математическая статистика, вычислительная математика и т.п.

Третий этап технологического процесса подготовки решения задач ЭВМ представляет собой *алгоритмизацию* ее решения, т.е. разработку оригинального или адаптацию (уточнение и корректировку) уже известного алгоритма.

Алгоритмизация - это сложный творческий процесс. В основу процесса алгоритмизации положено фундаментальное понятие математики и программирования - алгоритм. Алгоритм - это конечный набор правил, однозначно раскрывающих содержание и последовательность выполнения операций для систематического решения определенного класса задач за конечное число шагов.

Любой алгоритм обладает следующими важными свойствами: Детерминированностью, массовостью, результатностью и дискретностью.

Детерминированность алгоритма (определенность, однозначность) - свойство, определяющее однозначность результата работы алгоритма при одних и тех же исходных данных. Это означает, что набор указаний алгоритма должен быть однозначно и точно понят любым исполнителем.

Массовость алгоритма - свойство, определяющее пригодность использования алгоритма для решения множества задач данного класса. Оно предполагает возможность варьирования исходными данными в определенных пределах. Свойство массовости алгоритма является определяющим фактором, обеспечивающим экономическую эффективность решения задач на ЭВМ, так как для задач, решение которых осуществляется один раз, целесообразность использования ЭВМ, как правило, диктуется внеэкономическими категориями.

Результатность алгоритма - свойство, означающее, что для любых допустимых исходных данных он должен через конечное число шагов (или итераций) завершить работу.

Дискретность алгоритма - свойство, означающее возможность разбиения определенного алгоритмического процесса на отдельные элементарные действия.

Таким образом, алгоритм дает возможность чисто механически решать любую задачу из некоторого класса однотипных задач.

Составление (адаптация) программ (кодирование) является завершающим этапом технологического процесса разработки программных средств. Он предшествует началу непосредственно машинной реализации алгоритма решения задачи. Процесс кодирования заключается в переводе описания алгоритма на один из доступных для ЭВМ языков программирования. В процессе составления программы для ЭВМ конкретизируются тип и структура используемых данных, а последовательность действий, реализующих алгоритм, отражается посредством конкретного языка программирования.

Этап тестирования и отладки. Оба эти процесса функционально связаны между собой, хотя их цели несколько отличаются друг от друга.

Тестирование представляет собой совокупность действий, назначенных для демонстрации правильности работы программы в заданных диапазонах изменения внешних условий и режимов эксплуатации программы. Цель тестирования заключается в демонстрации отсутствия (или выявлении) ошибок в разработанных программах на заранее подготовленном наборе контрольных примеров.

Процессу тестирования сопутствует понятие "отладка", которое подразумевает совокупность действий, направленных на устранение ошибок в программах, начиная с момента обнаружения фактов ошибочной работы программы и завершая устранением причин их возникновения.

По своему характеру (причине возникновения) ошибки в программах делятся на синтаксические и логические.

Синтаксические ошибки в программе представляют собой некорректную запись отдельных языковых конструкций с точки зрения правил их представления для выбранного языка программирования. (ошибки выявляются автоматически)

Далее проверяется логика работы программы на исходных данных. При этом возможны следующие основные формы проявления логических ошибок:

- в какой-то момент программа не может продолжать работу (возникает программное прерывание, обычно сопровождающееся указанием места в программе, где оно произошло);
- программа работает, но не выдает всех запланированных результатов и не выходит на останов (происходит ее "зацикливание");
- программа выдает результаты и завершает свою работу, но они полностью или частично не совпадают с контрольными.

После выявления логических ошибок и устранения причин их возникновения в программу вносятся соответствующие исправления и отладка продолжается.

Программа считается отлаженной, если она безошибочно выполняется на достаточно представительном наборе тестовых данных, обеспечивающих проверку всех ее участков (ветвей).

Процесс тестирования и отладки программ имеет итерационный характер и считается одним из наиболее трудоемких этапов процесса разработки программ. По оценкам специалистов, он может составлять от 30 до 50% в общей структуре затрат времени на разработку проектов и зависит от объема и логической сложности разрабатываемы программных комплексов.

Для сокращения затрат на проведение тестирования и отладки в настоящее время широко применяются специальные программные средства тестирования (например, генераторы тестовых данных) и приемы отладки (например, метод трассировки программ, позволяющий выявлять, все ли ветви программы были задействованы при решении задачи с заданными наборами исходных данных).

После завершения процесса тестирования и отладки программные средства вместе с сопроводительной документацией передаются пользователю *для эксплуатации.*

Основное назначение сопроводительной документации - обеспечить пользователя необходимыми инструктивными материалами по работе с программными средствами. Состав сопроводительной документации обычно оговаривается заказчиком (пользователем) и разработчиком на этапе подготовки технического задания на программное средство. Как правило, это документы, регламентирующие работу пользователя в процессе эксплуатации программы, а также содержащие информацию о программе, необходимую в случае возникновения потребности внесения изменений и дополнений в нее.

При передаче пользователю разработанных прикладных программных средств создается специальная комиссия, включающая в свой состав представителей разработчиков и заказчиков (пользователей). Комиссия в соответствии с заранее составленным и утвержденным обеими сторонами планом проводит работы *по приемке-передаче программных средств* и сопроводительной документации. По завершении работы комиссии оформляется акт приемки-передачи.

В процессе внедрения и эксплуатации прикладных программных средств могут выявляться различного рода ошибки, не обнаруженные разработчиком при тестировании и отладке программных средств. Поэтому при реализации достаточно сложных и ответственных программных комплексов по согласованию пользователя (заказчика) с разработчиком *этап эксплуатации* программных средств может быть разбит на два подэтапа: *экспериментальная (опытная)* и *промышленная эксплуатация*.

Смысл экспериментальной эксплуатации заключается во внедрении разработанных программных средств на объекте заказчика с целью проверки их работоспособности и удоб-

ства работы пользователей при решении реальных задач в течение достаточно длительного периода времени (обычно не менее года) Только после завершения периода экспериментальной эксплуатации и устранения выявленных при этом ошибок и учета замечаний программное средство передается в промышленную эксплуатацию.

Для повышения качества работ, оперативности исправления ошибок, выявляемых в процессе эксплуатации программных средств, также выполнения различного рода модификаций, в которых может возникнуть необходимость в ходе эксплуатации, разработчик может по договоренности с пользователем осуществлять их сопровождение.

Описанная схема технологического процесса разработки прикладных программных средств отражает их "жизненный цикл", т.е. временной интервал с момента зарождения программы до момента полного отказа от ее эксплуатации.

11.2 Современные методы разработки ПО

Метод нисходящего проектирования (метод пошаговой детализации, метод иерархического проектирования, top-down-подход)

Суть метода заключается в определении спецификаций компонентов системы путем последовательного выделения в ее составе отдельных составляющих и их постепенной детализации до уровня, обеспечивающего однозначное понимание того, что и как необходимо разрабатывать и реализовывать.

Этот метод является незаменимым при разработке сложных по характеру и больших по объему программ, когда к их разработке необходимо привлекать большое число программистов, работающих параллельно. Он позволяет концентрировать внимание разработчиков на наиболее ответственных частях программы, а также облегчает возможность постоянного контроля за ее работоспособностью по мере разработки, отладки и объединения отдельных составляющих программ за счет организации непрерывности этого процесса в течение всей разработки.

Для ускорения разработки программного комплекса часто вместо некоторых программ нижнего уровня, находящихся в процессе разработки, могут применяться специальные "программы-заглушки" Программы-заглушки требуются только на ранних стадиях разработки для того, чтобы не сдерживать общий ход создания программного комплекса. Суть программы-заглушки заключается в том, что при обращении к ней в соответствии с заданным набором исходных тестовых данных она не формирует, а выбирает результат "решения" из заранее подготовленного набора. Благодаря этому обеспечивается возможность имитировать работу на ЭВМ реально создаваемой программы, а следовательно, осуществлять проверку работоспособности программ верхнего уровня еще до того, как будут разработаны и отлажены все составляющие программы нижнего уровня.

Модульное проектирование

Реализация метода *нисходящего проектирования* тесно связана с другим понятием программирования - *модульным проектированием*, так как на практике при декомпозиции сложной программы возникает вопрос о разумном пределе ее дробления на составные части. Вместе с тем понятие модульности нельзя сводить только к представлению сложных программных комплексов в виде набора отдельных функциональных блоков.

Модуль - это последовательность логически взаимосвязанных фрагментов задачи, оформленных как отдельная часть программы. При этом программные модули должны обладать следующими свойствами:

- на модуль можно ссылаться (т.е. обращаться к нему) по имени, в том числе и из других модулей;
- по завершении работы модуль должен возвращать управление тому модулю, который его вызывал;

- модуль должен иметь один вход и выход;
- модуль должен иметь небольшой размер, обеспечивающий его обозримость.

При разработке сложных программ в них выделяют головной управляющий модуль, подчиненные ему модули, обеспечивающие реализацию отдельных функций управления, функциональную обработку (т.е. непосредственную реализацию основного назначения программного комплекса), а также вспомогательные модули, обеспечивающие сервисное обслуживание пакета (например, сбор и анализ статистики работы программы, обработка различного рода ошибочных ситуаций, обучение и выдача подсказок и т.п.).

Модульный принцип разработки программ обладает следующими преимуществами:

- большую программу могут разрабатывать одновременно несколько исполнителей, и это позволяет сократить сроки ее разработки;
- появляется возможность создавать и многократно использовать в дальнейшем библиотеки наиболее употребимых программ;
- упрощается процедура загрузки больших программ в оперативную память, когда требуется ее сегментация;
- возникает много естественных контрольных точек для наблюдения за осуществлением хода разработки программ, а в последующем для контроля за ходом исполнения программ;
- обеспечивается более эффективное тестирование программ, проще осуществляются проектирование и последующая отладка.

Преимущества модульного принципа построения программ особенно наглядно проявляются на этапе сопровождения и модификации программных продуктов, позволяя значительно сократить затраты сил и средств на реализацию этого этапа.

Структурное программирование

Актуальная для начального периода развития и использования ЭВМ проблема разработки программ, занимающих минимум основной памяти и выполняющихся за кратчайшее время, в последующем в связи резким падением стоимости аппаратной части ЭВМ, значительным возрастанием их быстродействия и объемов памяти сменилась необходимостью разработки и применения принципиально новых методов составления программ. Все это нашло свое воплощение в разработке принципа *структурного программирования.* Одной из целей структурного программирования было стремление облегчить разработку и отладку программных модулей, а главное - их последующее сопровождение и модификацию.

В настоящее время структурное программирование - это целая дисциплина, объединяющая несколько взаимосвязанных способов создания ясных, легких для понимания программ. Эффективность применения современных универсальных языков программирования во многом определяется удобством написания с их помощью структурных программ.

CASE-технологии

За последнее десятилетие в области средств автоматизации программирования сформировалось новое направление под общим названием CASE-технологии (Computer Aided Software Engineering).

CASE-технология представляет собой совокупность средств системного анализа, проектирования, разработки и сопровождения сложных программных систем, поддерживаемых комплексом взаимоувязанных инструментальных средств автоматизации всех этапов разработки программ. Благодаря структурным методам CASE-технология на стадиях анализа и проектирования обеспечивает разработчиков широкими возможностями для различного рода моделирования, а централизованное хранение всей необходимой для проектирования информации и контроль за целостностью данных гарантируют согласованность взаимодействия всех специалистов, занятых в разработке ПО.

Технологии RAD

В начале 80-х годов появилась методология, по которой разработка программы начиналась не после завершения процесса выработки окончательных требований к ней, а как только устанавливались требования на первый, “стартовый” (пилотный) вариант прикладной программы, позволяющий начать содержательную работу по ее реализации на компьютере.

Это дало пользователю возможность, получая уже с первых шагов конкретное представление о характере реализации задачи, уточнять ее постановку. Тем самым облегчался процесс экспериментального поиска нужного решения автоматизации задачи. Благодаря тесному взаимодействию разработчика с заказчиком (пользователем) на самом ответственном этапе создания прикладных программ между ними достигалось быстрое взаимопонимание цели поставленной задачи и возможности ее автоматизации в данных конкретных условиях. Это повышало скорость разработки программ и послужило основанием для названия такой технологии *RAD (Rapid Application Development* - быстрая разработка программ), которая получила широкое распространение.

Data Warehouse

Другое направление разработки прикладных программных средств, олицетворяющее собой современный подход к реализации широкого круга задач для принятия управленческих решений, базируется на концепции создания специального хранилища данных (Data Warehouse). Основное отличие концепции Data Warehouse от традиционного представления баз данных заключается в следующем:

- во-первых, в том, что актуализация данных в Data Warehouse означает не обновление элементов информации, а добавление новых элементов к уже имеющимся (что расширяет возможности проведения различного рода сравнительного анализа);
- во-вторых, в том, что наряду с информацией, непосредственно отражающей состояние системы управления, в Data Warehouse аккумулируются и метаданные.

Метаданные (данные о данных) облегчают возможность визуального представления содержимого Data Warehouse, позволяют, "перемещаясь" по хранилищу, быстро отбирать необходимые данные для последующей обработки.

Основные типы метаданных Data Warehouse отражают:

- структуру и содержимое хранилища;
- соответствие между исходными и выходными данными;
- объемные характеристики данных;
- критерии архивирования;
- отношения между данными;
- информацию по кодированию;
- интервал жизни данных и т.п.

Концепция *Data Warehouse* поддерживается RAD средствами разработки прикладного ПО.

Концепция Data Warehouse обеспечивает возможность разработки программных приложений для поддержки процессов принятия решений с использованием OLAP-систем.

Система OLAP (On-Line Analytical Process) предоставляет возможность разработки информационных систем, ориентированных на уна организацию многомерных баз данных и создание корпоративных сетей, а также обеспечивает поддержку Web-технологий в сетях Internet/Intranet

Успешное применение инструментальных средств OLAP-систем объясняется быстротой разработки приложений, гибкостью и широкими возможностями в области доступа к данным и их преобразования. В настоящее время на рынке ПО предлагается большое число OLAP-стем, разработчиками которых являются различные фирмы, например IBM, Informix, Microsoft, Oracle, Sybase и др.

11.3 Инструментарий технологии программирования

Инструментарий технологии программирования - программные продукты поддержки (обеспечения) технологии программирования.

В рамках этого направления сформировались следующие группы программных продуктов (рис. 10.2):

1) средства для создания приложений, включающие:

- локальные средства, обеспечивающие выполнение отдельных работ по созданию программ;
- интегрированные среды разработчиков программ, обеспечивающие выполнение комплекса взаимосвязанных работ по созданию программ;

2) средства для создания информационных систем (CASE- технология), представляющие методы анализа, проектирования и создания программных систем и предназначенные для автоматизации процессов разработки и реализации информационных систем.

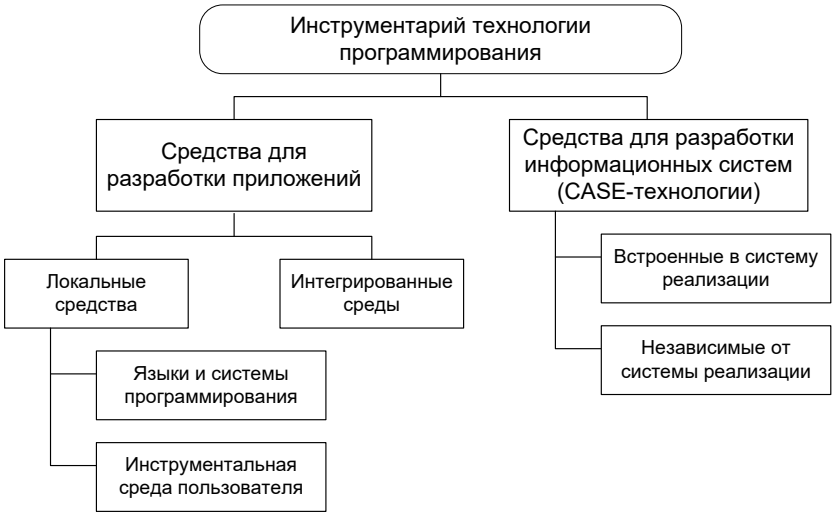


Рис. 11.2 - Классификация инструментария технологии программирования

11.4 Средства для создания приложений

Локальные средства разработки программ.

Эти средства на рынке программных продуктов наиболее представительны и включают языки и системы программирования, а также инструментальную среду пользователя.

Язык программирования - формализованный язык для описания алгоритма решения задачи на компьютере.

Средства для создания приложений - совокупность языков и систем программирования, а также различные программные комплексы для отладки и поддержки создаваемых программ.

Языки программирования можно условно разделить на следующие классы (если в качестве признака классификации взять синтаксис образования конструкций языка):

- машинные языки (computer language) - языки программирования, воспринимаемые аппаратной частью компьютера (машинные коды);
- машинно-ориентированные языки (computer-oriented language) - языки программирования, которые отражают структуру конкретного типа компьютера (ассемблеры);
- алгоритмические языки (algorithmic language) - языки программирования, не зависящие от архитектуры компьютера (Паскаль, Си, Фортран, Бейсик и др.);

- процедурно-ориентированные языки (procedure-oriented language) - языки программирования, где имеется возможность написания программы как совокупности процедур (подпрограмм);
- проблемно-ориентированные языки (universal programming language) - языки программирования, предназначенные для решения задач определенного класса (Лисп, Пролог, Симула и др.);
- интегрированные системы программирования.

Другой классификацией языков программирования является их деление на языки, ориентированные на реализацию основ структурного программирования, и объектно-ориентированные языки, поддерживающие понятие объектов и их свойств и методов обработки.

Программа, подготовленная на языке программирования, проходит этап трансляции, когда происходит преобразование исходного кода программы (source code) в объектный код (object code), который далее пригоден к обработке редактором связей. Редактор связей специальная программа, обеспечивающая построение загрузочного модуля (load module), пригодного к выполнению (рис. 10.3).

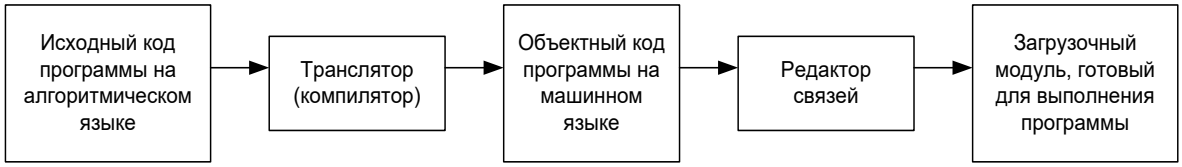


Рис. 11.3 - Схема процесса создания загрузочного модуля программы

Трансляция может выполняться с использованием средств компиляторов (compiler) или интерпретаторов (interpreter). Компиляторы транслируют всю программу, но без ее выполнения. Интерпретаторы, в отличие от компиляторов, выполняют пооператорную обработку и выполнение программы.

Существуют специальные программы, предназначенные для трассировки и анализа выполнения программ, так называемые отладчики (debugger). Лучшие отладчики позволяют осуществить трассировку (отслеживание выполнения программы в пооператорном варианте), идентификацию места и вида ошибок в программе, наблюдение за изменением значений переменных, выражений и т.п. Для отладки и тестирования правильности работы программ создается база данных контрольного примера.

Более мощным средством разработки программ являются *системы программирования.*

Системы программирования (programming system) включают:

- компилятор;
- интегрированную среду разработчика программ;
- отладчик;
- средства оптимизации кода программ;
- набор библиотек (возможно с исходными текстами программ);
- редактор связей;
- сервисные средства (утилиты) для работы с библиотеками текстовыми и двоичными файлами;
- справочные системы;
- документатор исходного кода программы;
- систему поддержки и управления проектом программного комплекса.

Средства поддержки проектов - новый класс средств разработки программного обеспечения, предназначенный для:

- отслеживания изменений, выполненных разработчиками программ;

- поддержки версий программы с автоматической разноской изменений;
- получения статистики о ходе работ проекта.

Инструментальная среда пользователя представлена специальными средствами, встроенными в пакеты прикладных программ, такими, как:

- библиотека функций, процедур, объектов и методов обработки;
- макрокоманды;
- клавишные макросы; языковые макросы;
- программные модули-вставки; конструкторы экранных форм и отчетов;
- генераторы приложений; языки запросов высокого уровня;
- языки манипулирования данными; конструкторы меню и многое другое.

Средства отладки и тестирования программ предназначены для подготовки разработанной программы к промышленной эксплуатации.

Интегрированные среды разработки программ

Дальнейшим развитием локальных средств разработки программ, являются интегрированные программные среды разработчиков.

Основное назначение инструментария данного вида - повышение производительности труда программистов, автоматизация создания кодов программ, обеспечивающих интерфейс пользователя графического типа, разработка приложений для архитектуры клиент-сервер, запросов и отчетов.

11.5 CASE-технологии

CASE-технологии - относительно новое направление, формировавшееся на рубеже 80-х годов.

CASE-технологии делятся на две группы:

- встроенные в систему реализации, в которых все решения по проектированию и реализации привязаны к выбранной системе явления базами данных (СУБД);
- независимые от системы реализации, в которых все решения по проектированию ориентированы на унификацию начальных этапов жизненного цикла, средств их документирования и обеспечивают большую гибкость в выборе средств реализации.

Основное достоинство CASE-технологии - поддержка коллективной работы над проектом за счет возможности работы в локальной сети разработчиков, экспорта/импорта любых фрагментов проекта, организационного управления проектом.

Некоторые CASE-технологии ориентированы только на системных проектировщиков и предоставляют специальные графические средства для изображения различного вида моделей:

- диаграмм потоков данных (DFD - data flow diagrams) совместно со словарями данных и спецификациями процессов;
- диаграмм "сущность-связь" (ERD - entity relationship diagrams), являющихся информационной моделью предметной области;
- диаграмм переходов состояний (STD - state transition diagrams), учитывающих события и реакцию на них системы обработки данных.

Диаграммы DFD устанавливают связь источников информации с потребителями, выделяют логические функции (процессы) образования информации, определяют группы элементов данных и их хранилища (базы данных).

Описание структуры потоков данных, определение их компонентов хранятся в актуальном состоянии в словаре данных, который выступает как база данных проекта. Каждая логическая функция может детализироваться с помощью DFD нижнего уровня согласно методам исходящего проектирования.

Этими CASE-технологиями выполняются автоматизированное проектирование спецификаций программ (задание основных характеристик для разработки программ) и ведение словаря данных.

Другой класс CASE-технологий поддерживает только разработку программ, включая:

- автоматическую генерацию кодов программ на основании их спецификаций;
- проверку корректности описания моделей данных и схем потоков данных;
- документирование программ согласно принятым стандартам и актуальному состоянию проекта;
- тестирование и отладку программ.

Кодогенерация программ выполняется двумя способами: создание каркаса программ и создание полного продукта. Каркас программы служит для последующего ручного варианта редактирования исходных текстов, обеспечивая возможность вмешательства программиста; полный продукт не редактируется вручную.

В рамках CASE-технологий проект сопровождается целиком, а не только его программные коды. Проектные материалы, подготовленные в CASE-технологии, служат заданием программистам, а само программирование скорее сводится к кодированию - переводу на определенный язык структур данных и методов их обработки, если не предусмотрена автоматическая кодогенерация.

11.6 Языки и системы программирования

Поколения языков программирования

Языки программирования принято делить на пять поколений.

В первое поколение входят языки, созданные в начале 50-х годов, когда только появились первые компьютеры. 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 возникли 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) 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 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.

Обзор языков программирования высокого уровня

Fortran (Фортран)

Это первый компилируемый язык, созданный в 50-е годы.

Программисты, разрабатывавшие программы исключительно на ассемблере, выражали серьезное сомнение в возможности появления высокопроизводительного языка высокого уровня, поэтому основным критерием при разработке компиляторов Фортрана являлась эффективность исполняемого кода. Хотя в Фортране впервые был реализован ряд важнейших понятий программирования, удобство создания программ было принесено в жертву возможности получения эффективного машинного кода. Однако для этого языка было создано огромное количество библиотек, начиная от статистических комплексов и заканчивая пакетами управления спутниками. Фортран продолжает активно использоваться во многих организациях. Имеется стандартная версия Фортрана HPF (High Performance Fortran) для параллельных суперкомпьютеров со множеством процессоров.

Cobol (Кобол).

Это компилируемый язык для применения в экономической области и решения бизнес - задач, разработанный в начале 60-х годов. Он отличается большой "многословностью" – его операторы иногда выглядят как обычные английские фразы. В Коболе были реализованы очень мощные средства работы с большими объемами данных, хранящимися на различных внешних носителях. На этом языке создано очень много приложений, которые эксплуатируются и сегодня.

Algol (Алгол). Компилируемый язык, созданный в 1960г. Он был призван заменить Фортран, но из-за более сложной структуры не получил широкого распространения. В 1968г. была создана версия Алгол 68, по своим возможностям и сегодня опережающая многие языки программирования, однако из-за отсутствия достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы.

Pascal (Паскаль)

Язык Паскаль, созданный в конце 70-х годов, во многом напоминает Алгол, но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

Basic (Бейсик)

Для этого языка имеются и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Он создавался в 60-х годах в качестве учебного языка и очень прост в изучении.

C (Си)

Данный язык был создан в лаборатории Bell и первоначально не рассматривался как массовый. Он планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы, и в то же время не зависеть от конкретного типа процессора. Си во многом похож на Паскаль и имеет дополнительные средства для прямой работы с памятью (указатели). На этом языке в 70-е годы написано множество прикладных и системных программ и ряд известных операционных систем (Unix).

C++ (Си++)

Си++ - это объектно-ориентированное расширение языка Си, разработан в 1980 г. В нем реализовано множество новых мощных возможностей, которые позволили резко повы-

сить производительность труда программистов, однако создание сложных и надежных программ требует от разработчиков профессиональной подготовки высокого уровня.

Java (Ява)

Этот язык был создан компанией Sun в начале 90-х годов на основе Си++. Он призван упростить разработку приложений на основе Си++ путем исключения из него всех низкоуровневых возможностей. Но главная особенность этого языка - компиляция не в машинный код, а в платформно-независимый байт-код (каждая команда занимает один байт). Этот байт-код может выполняться с помощью интерпретатора - виртуальной Java-машины JVM (Java Virtual Machine), версии которой созданы сегодня для любых платформ.

Пока недостаток этого языка - невысокое быстродействие.

Языки программирования баз данных

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:

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 фактически в каждой СУБД помимо поддержки языка SQL имеется свой уникальный язык, ориентированный на особенности этой СУБД и не переносимый на другие системы.

Сегодня в мире насчитывается пять ведущих производителей СУБД:

Microsoft (SQL Server), IBM (DB2), Oracle, Software AG (Adabas), Informix и Sybase. Их продукты нацелены на поддержку одновременной работы тысяч пользователей в сети, а базы данных могут храниться в распределенном виде на нескольких серверах.

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 похожие, но несовместимые версии языков семейства dBase реализованы в продуктах Visual FoxPro фирмы Microsoft и Visual dBase фирмы Inprise.

Языки программирования для Интернета

С активным развитием глобальной сети было создано немало 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 Web-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.

VRML. В 1994г. 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), вращать в любых направлениях, масштабировать, регулировать освещенность и т. д.

12. ОСНОВНЫЕ ОПЕРАТОРЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ PASCAL.

12.1 Алфавит языка Паскаль. Переменные. Типы переменных

Алфавит языка Паскаль.

Алфавит языка *- это конечный набор символов, которые используются при написании любой конструкции на языке.*

Алфавит языка Паскаль можно разбить на 3 группы :

- **буквы** строчные и прописные буквы латинского и русского алфавита;
- **цифры** - арабские 0..9

3. специальные символы - (+,-,*,/,=,<,>,<>,<=,>=,.:;{}’ := ( )),
служебные слова : and, begin,const,div,do,else,for, и т.д. и указывается тип переменных списка.

Переменная - это область памяти, названная собственным именем, которая может менять свое значение в процессе выполнения программы. Переменная характеризуется именем, типом и значением.

Константа - не изменяет своего значения в процессе выполнения программы, она может быть задана явно своим значением или обозначена именем.

Имя (идентификатор) переменной или константы задается латинскими буквами и арабскими цифрами. В качестве идентификатора нельзя использовать служебные слова. Идентификатор должен быть уникальным, т.е. в данном блоке программы один идентификатор не может быть использован для обозначения более чем одного объекта.
Например : Zap, gor,X, p1, summa, a28,rar_1, proba, x1,y1,max,min и т.д.

Типы переменных

Тип		Диапазон значений
Целый	ShortInt	-128...127
	Integer	-32768 ...32767
	Longint	-2147483648 ... 2147483647
	Byte	0 255
	Word	0 ... 65535
Вещественный	Real	$2{,}9\ 10^{-39}$..... $1{,}7\ 10^{38}$ 11..12 значащих цифр
	Single	7..8 значащих цифр
	Double	15..16 значащих цифр
	Extended	19..20 значащих цифр
	Comp	19..20 значащих цифр
Логический	Boolean	TRUE FALS
Символьный	Char	1 символ
Строковый	String	Массив символов, по умолчанию длина 256 симво-
Массивы	Array	
Записи	Record	

Множества	Set	
Файлы	File	
Указатели	Pointer	

11.2 Стандартные математические функции и правила записи арифметических выражений.

Стандартные математические функции:

abs(x) Абсолютное значение x, т.е. x

exp(x) Значением функции является e в степени x. sin(x) и cos(x)

Значение синуса или косинуса x, x должен задаваться в радианах.

arctan(x) Арктангенс x.

ln(x) Натуральный логарифм x (x>0)

sqr(x) Квадрат x.

sqrt(x) Квадратный корень из x.

random(x) Случайное число из диапазона

0<=...< x

Pi Значение пи.

odd(x) Значение функции true, если x нечетен, и false в противном случае.

inc(x,n) Значением является x увеличенное на n.

dec(x,n) Значением является x уменьшенное на n.

int(x) Целая часть числа x.

frac(x) Дробная часть числа x.

trunc(x) Целая часть в форме longint.

round(x) Значение x округленное до следующего целого числа.

Правила записи арифметических выражений.

1. Все данные, входящие в арифметическое выражение, должны быть одного типа. Допускается

использовать вместе данные целого и действительного типа.

2. Записывать все составные части в одну строку без подстрочных и надстрочных индексов.

3. Использовать скобки одного типа - круглые. ([{ и другие скобки применять запрещается)

4. Нельзя записывать подряд 2 знака арифметических операций.

5. Вычисления выполняются слева направо в соответствии со старшинством операций:

1) вычисление функций;

2) * / DIV (деление нацело)

MOD (получение остатка от деления) 3) + -

11.3 Условные операторы

Условные операторы предназначены для выбора к исполнению одного из возможных действий (операторов) в зависимости от некоторого условия (при этом одно из действий может быть пустым, т. е. отсутствовать). В качестве условий выбора используется значение логического выражения.

В Турбо Паскале имеются два условных оператора: *if и case.*

Оператор условия if

Оператор условия *if* является одним из самых популярных средств, изменяющих естественный порядок выполнения операторов программы.

Он может принимать одну из следующих форм:

- *if <условие> then <оператор1>*
  else <оператор2>;
- *if <условие> then <оператор>;*

В переводе с английского языка данные форматы можно определить как:

- ЕСЛИ<*условие*>ТО<*оператор1*>ИНАЧЕ<*оператор2*>
- ЕСЛИ<*условие*>ТО<*оператор*>

Оператор условия *if* выполняется следующим образом. Сначала вычисляется выражение, записанное в условии. В результате его вычисления получается значение булевского типа.

В первом случае, если значение выражения есть *True* (истина), выполняется *<оператор1>,* указанный после слова *then* (в переводе –“то”). Если результат вычисления выражения в условии есть *False* (ложь), то выполняется *<оператор2>*.

Во втором случае, если результат выражения *True*, выполняется *<оператор>,* если *False* - оператор, следующий сразу за оператором *if*. Операторы *if* могут быть *вложенными*.

Особенность применения оператора if. При использовании вложенных условных операторов может возникнуть синтаксическая неоднозначность, например:

if условие1 then if условие2 then <оператор1> else <оператор2>

Возникающая двусмысленность, к какому оператору *if* принадлежит часть *else <оператор2>,* разрешается тем, что служебное слово *else* всегда ассоциируется (связывается) с ближайшим по тексту служебным словом *if,* которое еще не связано со служебным словом *else*.

Оператор выбора case

Если один оператор *if* может обеспечить выбор из двух альтернатив, то оператор выбора *case* позволяет сделать выбор из произвольного числа имеющихся вариантов. Он состоит из выражения, называемого *селектором* (*selection — выбор альтернативы*), и *списка параметров*, каждому из которых предшествует список *констант выбора* (список может состоять и из одной константы).

Формат записи оператора case:

case <выражение-селектор> of
<список1>: <оператор1; >
<список2>: <оператор2; >
...
<списокN>: <операторN>
else <оператор>
end;

Селектор должен относиться к одному из целочисленных типов (находящихся в диапазоне — 32768..32767): булевскому, литерному или пользовательскому. Список констант выбора состоит из произвольного количества значений, или диапазонов, отделенных друг от друга запятыми. Границы диапазона записываются двумя константами через разграничитель "..". Тип констант в любом случае должен совпадать с типом селектора. В синтаксическом описании, приведенном выше, предполагается использование одного оператора для каждой альтернативы, но при необходимости можно задать несколько операторов, сгруппировав их в составной оператор. В то же время ветвь *else* допускает использование последовательности операторов, разделенных символом ";".

Правила использования оператора case. При использовании оператора выбора *case* должны выполняться следующие правила:

1. Значения выражения "переключателя", записанного после служебного слова *case*, должны принадлежать дискретному типу (лат. discretus — прерывистый, дробный, состоящий из отдельных частей); для целого типа они должны лежать в диапазоне *integer*.

2. Все константы, предшествующие операторам альтернатив, должны иметь тип, совместимый с типом выражения.

3. Все константы в альтернативах должны быть уникальны в пределах оператора варианта (т. е. повторения констант в альтернативах не допускаются); диапазоны не должны пересекаться и не должны содержать констант, указанных в данной или других альтернативах.

11.4 Операторы повтора

Если в программе возникает необходимость неоднократно выполнить некоторые операторы, то используются *операторы повтора (цикла).* В языке Паскаль различают три вида операторов цикла: *while, repeat, for*. Они используются для организации циклов различных типов. Выражение, управляющее повторениями, должно иметь булевский тип.

Если число повторений оператора (составного оператора) заранее неизвестно, а задано лишь условие его повторения (или окончания), используются операторы *while, repeat*. Оператор *for* используется, если число повторений заранее известно.

Оператор повтора for

В случаях, когда число повторений может быть заранее известно, для организации циклической обработки информации применяется оператор повтора *for*. Часто этот оператор повтора называют *оператором цикла с параметром,* так как число повторений задается переменной, называемой *параметром цикла*, или управляющей переменной. Оператор повтора for состоит из *заголовка* и *тела цикла*.

Он может быть представлен в двух форматах:

• *for <параметр цикла> := <S1> to <S2> do <оператор>;*

• *for <параметр цикла> := <S1> downto <S2> do <оператор>;*

где *Sl* и *S2* — выражения, определяющие соответственно начальное и конечное значения параметра цикла;

for ... do — заголовок цикла;

<оператор> — тело цикла.

Тело цикла может быть простым или составным оператором. Оператор *for* обеспечивает выполнение тела цикла до тех пор, пока не будут перебраны все значения параметра цикла от начального до конечного.

Заголовок оператора повтора *for* определяет:

• диапазон изменения значений управляющей переменной (параметра цикла) и одновременно число повторений оператора, содержащегося в теле цикла;

• направление изменения значения параметра цикла (возрастание — *to* или убывание—*downto*).

Шаг изменения управляющей переменной - единица.

Ограничения использования параметра цикла в операторе for. На использование управляющей переменной (параметра цикла) в цикле *for* налагаются следующие ограничения:

1. В качестве параметра должна использоваться простая переменная, описанная в текущем блоке.

2. Управляющая переменная должна иметь дискретный тип.

3. Начальные и конечные значения диапазона должны иметь тип, совместимый с типом управляющей переменной. При этом допустим любой скалярный тип, кроме вещественного.

4. В теле цикла запрещается явное изменение значения управляющей переменной (например, оператором присваивания).

5. После завершения оператора значение управляющей переменной становится неопределенным, если только выполнение оператора не было прервано оператором перехода.

Оператор повтора Repeat

Оператор повтора *repeat* имеет две особенности:

1. Условие проверяется после очередного выполнения операторов тела цикла (очередной итерации) и таким образом гарантируется хотя бы однократное выполнение цикла.

2. Критерием прекращения цикла является равенство выражения константе *True*.

За это цикл *repeat* часто называют *циклом с постусловием*, или *циклом "ДО",* так как он прекращает выполняться, как только значение выражения условия, записанного после слова *until,* равно *True (истина).*

Оператор повтора *repeat* состоит из *заголовка repeat, тела и условия окончания until.*

Формат записи:

repeat
<оператор;>
...
<оператор>
until <условие окончания цикла>;

Операторы, заключенные между словами *repeat и until,* являются телом цикла. Вначале выполняется тело цикла, затем проверяется условие выхода из цикла. Именно поэтому цикл, организованный с помощью оператора *repeat*, в любом случае *выполнится хотя бы один раз*. Если результат булевского выражения равен *False,* то тело цикла активизируется еще раз; если результат *True*, происходит выход из цикла.

При программировании операторов тела цикла следует обеспечить влияние, по крайней мере, одного из операторов тела цикла на значение условия, иначе цикл будет выполняться бесконечно.

Оператор повтора while

Оператор *while (пока)* часто называют *оператором цикла с предусловием* за то, что проверка условия выполнения тела цикла производится в самом начале оператора.

Формат записи:

while <условие продолжения повторений> do
<тело цикла>;

Условие - булевское выражение, *тело цикла* - простой или составной оператор.

Перед каждым выполнением тела цикла вычисляется значение выражения условия. Если результат равен *True,* тело цикла выполняется и снова вычисляется выражение условия. Если результат равен *False*, происходят выход из цикла и переход к первому после *while* оператору.

11.5 Массивы. Описание массивов

Рассмотренные ранее *простые типы данных* определяют различные множества атомарных (неразделимых) значений. В отличие от них *структурные типы данных* задают множества сложных значений, каждое из которых образует совокупность нескольких значений другого типа. В структурных типах выделяют регулярный тип (массивы).

С понятием "массив" приходится сталкиваться при решении научно-технических и экономических задач обработки совокупностей большого количества значений. В общем случае *массив* - это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип.

Название *регулярный тип* (или ряды) массивы получили за то, что в них объединены однотипные (логически однородные) элементы, упорядоченные по индексам, определяющим положение каждого элемента в массиве.

В качестве элементов массива можно использовать и любой другой ранее описанный тип, поэтому вполне правомерно существование массивов записей, массивов указателей, массивов строк, массивов и т.д. Элементами массива могут быть данные любого типа, включая структурированные. Тип элементов массива называется *базовым*. Особенностью языка Паскаль является то, что *число элементов массива фиксируется* при описании и в процессе выполнения программы не меняется.

Элементы, образующие массив, упорядочены таким образом, что каждому элементу соответствует совокупность номеров (индексов), определяющих его местоположение в общей последовательности. *Доступ* к каждому отдельному элементу осуществляется путем индексирования элементов массива. *Индексы* представляют собой выражения любого скалярного типа, кроме вещественного. Тип индекса определяет границы изменения значений индекса. Для описания массива предназначено словосочетание: *array of (массив из).*

Формат записи массивов:

Type
<имя типа> = array [тип индекса] of <тип компонента>;
Var
<идентификатор,..> : <имя типа>;

Массив может быть описан и без представления типа в разделе описания типов данных:

Var
<идентификатор,...> : array [тип индекса] of <тип компонента>;

Примеры описания одномерных и двумерных массивов

Если в описании массива задан один индекс, массив называется *одномерным*, если два индекса - *двумерным*, если n индексов — *n-мерным массивом*. Одномерный массив соответствует понятию линейной таблицы (вектора), двумерный - понятию прямоугольной таблицы (матрицы, набору векторов). Размерность ограничена только объемом памяти конкретного компьютера.

Одномерные массивы обычно используются для представления *векторов*, а *двумерные*- для представления *матриц*.

Элементы массива располагаются в памяти последовательно. Элементы с меньшими значениями индекса хранятся в более низких адресах памяти. Многомерные массивы располагаются таким образом, что самый правый индекс возрастает самым первым.

После объявления массива каждый его элемент можно обработать, указав *идентификатор (имя) массива* и *индекс элемента* в квадратных скобках. Например, запись *Mas[2]*, *Vector[10]* позволяет обратиться ко второму элементу массива *Mas* и десятому элементу массива *Vector*. При работе с *двумерным массивом* указываются два индекса, с n-мерным массивом - n индексов. Например, запись *Matr[4,4]* делает доступным для обработки значение элемента, находящегося в *четвертой строке четвертого столбца* массива *Matr*.

Паскаль не имеет средств ввода-вывода элементов массива сразу, поэтому ввод и вывод значений производится поэлементно. Значения элементам массива можно присвоить с помощью оператора присваивания, как показано в примере инициализации, однако чаще

всего они вводятся с экрана с помощью оператора *Read* или *Readln* с использованием оператора организации *цикла for*:

Перестановка значений элементов массива осуществляется с помощью дополнительной переменной того же типа, что и базовый тип массива.

11.6 Понятие структурного программирования. Подпрограммы.

Значительное увеличение сложности задач, решаемых с помощью ЭВМ, приводит к увеличению размеров и сложности программ, что порождает дополнительные трудности при их разработке и отладке. Увеличение продолжительности жизненного цикла программ приводит к тому, что с течением времени из-за изменения условий использования программ возникает необходимость их модификации, повышения их эффективности, удобства пользования ими.

Для разрешения возникших при этом проблем в практике программирования выработан ряд приемов и методов, которые принято называть *методами структурного программирования.*

Под *структурным программированием* понимают такие методы разработки и записи программы, которые ориентированы на максимальные удобства для восприятия и понимания ее человеком. При прочтении программы в ее следующих друг за другом фрагментах должна четко прослеживаться логика ее работы, т. е. не должно быть "скачков" на фрагменты программы, расположенные где-то в другом месте программы.

Структурное программирование — "программирование без go to", т. е. не используются операторы перехода без особой необходимости. В связи с этим отдельные фрагменты программы представляют собой некоторые логические (управляющие) структуры, которые определяют порядок выполнения содержащихся в них правил обработки данных. Любая программа получается построенной из стандартных логических структур, число типов которых невелико.

Основные логические структуры:

Следование — последовательность операторов, групп операторов, выполняемых друг за другом в порядке их следования в тексте программы.

Ветвление — управляющая структура, которая в зависимости от выполнения заданного условия определяет выбор для исполнения одного из двух или более заданных в этой структуре групп операторов.

Повторение — цикл, в котором группа операторов может выполняться повторно, если соблюдается заданное условие.

Технология нисходящего программирования базируется на методе программирования "сверху вниз". Часто этот метод *называют методом пошаговой детализации*. Основой такого метода является идея постепенной декомпозиции исходной задачи на ряд подзадач. Сначала формулируется самая грубая модель решения, отдельные детали которой на первом этапе могут быть довольно расплывчатыми. По мере разработки программы, разбивая наиболее неясные части алгоритма и добиваясь все более точных и детализированных формулировок получают более подробное решение, как бы опускаясь с большой высоты ниже и начиная при этом различать более мелкие детали. Решение отдельного фрагмента сложной задачи может представлять собой самостоятельный программный блок, называемый *подпрограммой.* Такой процесс детализации продолжается до тех пор, пока не станут ясны все детали решения задачи. В этом случае программу решения сложной задачи можно представить как иерархическую совокупность относительно самостоятельных фрагментов — подпрограмм.

Подпрограммой называют обособленную, оформленную в виде отдельной синтаксической конструкции и снабженную именем часть программы. Использование подпрограмм

позволяет, сосредоточив в них подробное описание некоторых операций, в остальной программе только указывать имена подпрограмм, чтобы выполнить эти операции. Такие вызовы подпрограммы возможны неоднократно из разных участков программы, причем при вызове подпрограмме можно передать некоторую информацию (различную в разных вызовах), чтобы одна и та же подпрограмма выполняла решение подзадачи для разных случаев.

11.7 Функции в Паскале

За наличие подпрограмм как средства структурирования программ язык программирования Турбо Паскаль называется *процедурно-ориентированным*.

Подпрограммы в Турбо Паскале реализованы посредством *процедур и функций*. Имея один и тот же смысл и аналогичную структуру, процедуры и функции различаются назначением и способом их использования.

Функция — это независимая именованная часть программы, которую можно вызвать по имени для выполнения определенных действий. Структура функции повторяет структуру программы.

Особенности использования функции:

- функция передает в точку вызова скалярное значение;
- имя функции может входить в выражение как операнд.

Например, функция *Chr(65)* возвратит в точку вызова символ А *(код ASCII — 65)*, *Sqr(X)* — возведет в квадрат значения целого или вещественного X и возвратит в точку вызова вычисленное значение квадрата числа X.

Все процедуры и функции языка Турбо Паскаль делятся на две группы: *встроенные (стандартные)* и *определенные пользователем*. Первые входят в состав языка и вызываются для выполнения по строго фиксированному имени. Вторые разрабатываются и именуются самим пользователем. Все стандартные средства расположены в *специализированных библиотечных модулях*, которые имеют системные имена.

11.8 Стандартные библиотечные модули

В систему Турбо Паскаль версии 6.0 и старше включены восемь модулей: *System, Crt, Dos, Graph, Graph3, Overlay, Printer, Турбо3* и специализированная библиотека *Турбо Vision.* Модуль *System* подключается *по умолчанию*, все остальные должен подключать программист с помощью зарезервированного слова *uses*.

Например:

Uses Crt, Dos, Printer;

Назначение каждого модуля:

System — содержащиеся в нем подпрограммы обеспечивают работу всех остальных модулей системы.

Crt — содержит средства управления дисплеем и клавиатурой компьютера.

Dos — включает средства, позволяющие реализовывать различные функции DOS.

Graph3 — поддерживает использование стандартных графических подпрограмм версии Турбо Паскаль 3.0.

Overlay — содержит средства организации оверлейных программ.

Printer — обеспечивает быстрый доступ к печатающему устройству.

Turbo3 — обеспечивает максимально возможную совместимость с версией Турбо Паскаль 3.0.

Graph — содержит пакет графических средств, обеспечивающих эффективную работу с адаптерами CGA, EGA, VGA, IBM 3270 и т.д.

Турбо Vision— библиотека объектно-ориентированных подпрограмм для разработки пользовательских интерфейсов.

11.9 Встроенные функции и процедуры

Модуль *System* подключается к программе автоматически, поэтому его имя не указывается в разделе *Uses*. По этой причине программе становятся доступны его встроенные процедуры и функции.

Арифметические процедуры и функции

Abs(X:real/integer): real/integer — вычисление абсолютной величины X. Тип результата совпадает с типом параметра.

ArcTan(X:real):real — вычисление угла, тангенс которого равен X радиан.

Cos(X:real) : real — вычисление косинуса X, параметр задает значение угла в радианах.

Exp(X:real) : real — вычисление экспоненты X, т.е. значение E в степени X. E является основанием натурального логарифма и равно 2.718282.

Frac(X:real):real — вычисление дробной части X.

Int(X:real):real — вычисление целой части X.

Ln(X:real):real — вычисление натурального логарифма X, т, е. логарифма по основанию e (e = 2.718282).

Pi:real — возвращает значение числа Пи (3.141592653897932385).

Sin(X:real):real — вычисление синуса X. Параметр задает значение угла в радианах.

Sqr(X) — возведение в квадрат значения целого или вещественного значения X. Тип результата совпадает с типом параметра.

Sqrt(X:real):real — вычисление квадратного корня из X.

Random:real — генерирует значение случайного числа из диапазона 0..0.99.

Random(I:word):word — генерирует значение случайного числа из диапазона 0..I.

Randomize — изменение базы генератора случайных чисел.

Скалярные процедуры и функции

Dec(X{,n}) — процедура уменьшает значение целочисленной переменной X на величину n. При отсутствии необязательного параметра n значение X уменьшается на единицу.

Inc(X{,n}) — процедура увеличивает значение целочисленной переменной X на n. При отсутствии необязательного параметра n значение X увеличивается на единицу.

Pred(S) — функция возвращает элемент, предшествующий S в списке значений типа. Тид результата совпадает с типом параметра. Если предшествующего S элемента не существует, возникает программное прерывание.

Succ(S) — функция возвращает значение, следующее за S в списке значений типа. Тип результата совпадает с типом параметра. Если следующее за S значение отсутствует, возникает программное прерывание.

Odd(I: integer): boolean — возвращает True, если I нечетное, и False, если I четное.

Функции преобразования типов

Chr(I:byte):char — возвращает символ стандартного кода обмена информацией с номером, равным значению I. Если значение параметра больше 255, возникает программное прерывание.

Ord(S):longint — возвращает порядковый номер значения S в множестве, определенном типом S.

Round(X:real):longint — возвращает значение X, округленное до ближайшего целого числа.

Trunc(X:real):longint — возвращает ближайшее целое число, меньшее или равное X, если X >= 0, и большее или равное X, если X< 0.

Процедуры управления программой

Delay(I:word) — задержка выполнения программы на I мс.

Exit — выход из выполняемого блока в окружающую среду. Если текущий блок является процедурой или функцией, выход производится во внешний блок. Если Exit указана в операторной части основной программы, программа прекращает работу, и управление передается системе программирования.

Halt(N:word) — прекращение выполнения программы и передача управления системе программирования (если выполнялся .PAS-файл) или DOS (если выполнялся .EXE-файл). N — код завершения программы, передаваемый в операционную систему.

RunError(ErrCode:word) — прекращение выполнения программы и генерация ошибки времени выполнения. ErrCode — параметр типа byte, содержащий номер ошибки.

Специальные процедуры и функции

FillChar(P,Dl,Z) — заполняет побайтно область основной памяти заданным значением (заполнителем). Является одной из самых быстродействующих процедур. Область начинается с первого байта указанной переменной P и имеет размер, заданный параметром Dl. P — переменная любого типа; Dl — целочисленное выражение, указывающее длину; Z — заполнитель, выражение литерного или байтового типа.

Hi(I:integer):byte — выделяет старший байт значения I и помещает его в младший байт результата. Старший байт результата равен 0.

Lo(I: integer): byte — выделяет младший байт значения I и помещает его в младший байт результата. Старший байт результата равен 0.

SizeOf(IT:integer):word — вычисляет объем основной памяти в байтах, которую занимает указанная переменная или тип. IT — идентификатор переменной или типа данных.

Swap(I: integer): integer — обменивает содержимое младшего и старшего байтов целочисленного выражения, заданного параметром I типа integer.

11.10 Вызов стандартной процедуры или функции

Для использования стандартной процедуры или функции к программе подключается тот или иной специализированный библиотечный модуль, в котором записана данная стандартная процедура или функция (исключение составляет модуль *System*, так как он подключается к программе автоматически), для чего имя специализированного библиотечного модуля указывается в разделе *uses*. Затем в программе записывается вызов процедуры или функции, для чего записывается ее имя и указываются фактические параметры, например: *Pi, Sin(X), Chr(125), Inc(X,5).* Так как после выполнения функции ее значение присваивается имени, то имя функции используется в выражении.

11.11 Функции и процедуры пользователя

Функция, определенная пользователем, состоит из *заголовка* и тела *функции*.

Заголовок содержит зарезервированное слово *function,* идентификатор (имя) функции, заключенный в круглые скобки, необязательный список формальных параметров и тип возвращаемого функцией значения. Тело функции представляет собой локальный блок, по структуре аналогичный программе. В целом структура функции, определенной пользователем имеет вид:

function <имя> (Формальные параметры) : <тип результата>;
const ...
type ...
var
begin

<операторы>
end;

В разделе операторов должен находиться, по крайней мере, один оператор, *присваивающий имени функции значение*. В точку вызова возвращается результат последнего присваивания.

Обращение к функции осуществляется по имени с необязательным указанием списка аргументов. Каждый аргумент должен соответствовать формальным параметрам, указанным в заголовке, и иметь тот же тип.

Описание процедуры включает *заголовок (имя)* и *тело процедуры.* Заголовок состоит из зарезервированного слова *procedure*, идентификатора (имени) процедуры и необязательного, заключенного в круглые скобки, списка формальных параметров с указанием типа каждого параметра. *Имя процедуры* — идентификатор, уникальный в пределах программы. *Тело процедуры* представляет собой локальный блок, по структуре аналогичный программе.

Описания меток, констант, типов и т. д. действительны только в пределах данной процедуры. В теле процедуры можно использовать любые глобальные константы и переменные. Общая структура описания процедур:

procedure <имя> (Формальные параметры);
const ... ;
type . . . ;
var . . . ;
begin
<операторы>
end ;

11.11 Механизм передачи параметров

Как было показано в приведенных выше примерах программ с использованием процедур и функций, в заголовке процедуры или функции может быть задан список параметров, которые называются *формальными.* Название "формальные" эти параметры получили в связи с тем, что в этом списке заданы только имена для обозначения исходных данных и результатов работы процедуры, а при вызове подпрограммы на их место будут подставлены *конкретные значения* (выражений) и имен. Этот список указывается после имени подпрограммы и заключается в круглые скобки.

В списке формальных параметров должны быть перечислены имена формальных параметров и их типы. Имя параметра отделяется от типа двоеточием, а параметры друг от друга - точкой с запятой. Имена параметров одного типа можно объединять в подсписки, в которых имена отделяются друг от друга запятой.

Между формальными и фактическими параметрами должно быть полное соответствие:

• формальных и фактических параметров должно быть одинаковое количество

• порядок следования фактических и формальных параметров должен быть один и тот же;

• тип каждого фактического параметра должен совпадать с типом соответствующего формального параметра.

Параметры-значения. Параметры-значения используются только для передачи исходных данных из основной программы в подпрограмму (процедуру или функцию), в списке формальных параметров они перечисляются через запятую с обязательным указанием их типов, как было, например, в выше приведенных примерах:

procedure Abort(Msg: string);

function Step(N : integer; X : real): real;

Если формальный параметр объявлен как параметр-значение, то фактическим параметром может быть произвольное выражение. При вызове подпрограммы фактические параметры вычисляются и используются как начальные значения формальных параметров, т. е. осуществляется подстановка значений. Если формальный параметр определен как параметр-значение, то перед вызовом процедуры это значение вычисляется, полученный результат помещается во временную память и передается процедуре. Даже если фактический параметр - простейшее выражение в виде константы или переменной, все равно процедуре будет передана лишь копия этой константы (переменной). В процессе выполнения подпрограммы формальные параметры могут изменяться, но это никак не отразится на соответствующих фактических параметрах-переменных, которые сохранят те значения, которые имели до вызова подпрограммы, так как меняются не они, а их копия. Поэтому параметры-значения нельзя использовать для передачи результатов из подпрограммы в основную программу.

11.13 Стандартные строковые процедуры и функции

Строка — это последовательность символов кодовой таблицы персонального компьютера. При использовании в выражениях строка заключается в апострофы. Количество символов в строке (длина строки) может динамически изменяться от 0 до 255. Для определения данных строкового типа используется идентификатор *String*, за которым следует заключенное в квадратные скобки значение максимально допустимой длины строки данного типа. Если это значение не указывается, то по умолчанию длина строки равна 255 байт.

Переменную строкового типа можно определить через описание типа в разделе определения типов или непосредственно в разделе описания переменных. Строковые данные могут использоваться в программе также в качестве констант.

Недопустимо применение строковых переменных в качестве *селектора* в операторе *Case*.

Определение строкового типа устанавливает максимальное количество символов, которое может содержать строка.

Формат описания строкового типа

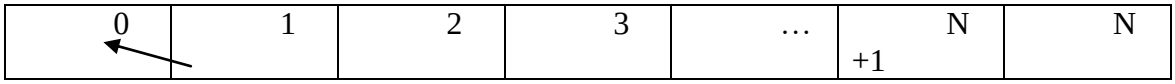
Type
<имя типа> =String [максимальная длина строки];
Var
<идентификатор, . . . > : <имя типа>;

Переменную типа *String* можно задать и без описания типа:

Var
<идентификатор, . . . > : String [максимальная длина строки];

Строка в языке Турбо Паскаль трактуется как цепочка символов. (Для строки из N символов отводится N+1 байт; N байт-для хранения символов строки, а один байт — для значения текущей длины строки).

К любому символу в строке можно обратиться, указав его номер. В самом начале строки (под нулевым номером) расположен байт, содержащий значение текущей длины строки.



текущая длина строки

Поэтому для определения объема памяти в байтах, требуемой для размещения строки, к значению ее максимальной длины прибавляется 1.

Например, для размещения в памяти переменных *Fstr, St1, St2* требуется соответственно 126, 35 и 51 байт (см. приведенный выше пример).

Рассмотрим структуру размещения строки в памяти на следующем примере. Пусть M — максимальная длина строки, L — текущая длина, A — ячейка памяти.

Тогда:

A —содержит величину текущей длины;
A+1 — первый символ строки;
A+L — последний значащий символ;
A+L+1
… — незанятые ячейки памяти
A+M

Стандартные строковые процедуры и функции

В целом, работу со стандартными строковыми процедурами и функциями можно свести в следующую таблицу:

Процеду-ра/Функция	*Описание*
Delete (St,Poz,N)	Удаление N символов строки St, начиная с позиции Poz
Insert (Str1,Str2,Poz)	Вставка строки Str1 в строку Str2, начиная с позиции Poz
Str (IBR, St)	Преобразование числового значения величины IBR и помещение результата в строку St
Val (St,IBR,Cod)	Преобразует значение St в величину целочисленного или вещественного типа и помещает результат в IBR. Значение St не должно содержать незначащих пробелов в начале и в конце. Cod - целочисленная переменная, в которой содержится 0, если ошибки не обнаружено, и номер позиции первого ошибочного символа, если произошла ошибка. Значение IBR в этом случае не определено.
Copy (St,Poz,N)	Выделяет из строки St подстроку длиной N символов, начиная с позиции Poz.Poz N-целочисленные выражения
Concat (Str1,Str2,…,StrN)	Выполняет сцепление строк Str1,Str2,…,StrN в том порядке, в каком они указаны в списке параметров.
Length (St)	Вычисляет текущую длину в символах строки St.Результат имеет целочисленный тип.
Pos(Str1,Str2)	Обнаруживает первое появление в строке Str2 подстроки Str1. Результат имеет целочисленный тип и равен номеру той позиции, где находится первый символ подстроки Str1.Если в Str2 подстроки Str1 не найдено, то результат равен нулю.
UpCase (Ch)	Преобразует строчную букву в прописную. Параметр и результат имеют литерный тип.

11.14 Структура и функции работы с записями

При решении научно- технических и экономических задач обработки совокупностей большого количества значений используются *массивы*. Но при работе с массивами основное ограничение заключается в том, что каждый элемент массива должен иметь *один и тот же тип данных*.

Иногда для решения задач, в которых возникает необходимость хранить и обрабатывать совокупности данных различного типа, используются записи.

Запись представляет собой наиболее общий и гибкий структурированный тип данных, так как она может быть образована из не однотипных компонентов и в ней явным образом выражена связь между элементами данных, характеризующими реальный объект.

Запись-это структурированный тип данных, состоящий из фиксированного числа компонентов, называемых *полями записи*. Определение типа записи начинается идентификатором *Record* и заканчивается зарезервированным словом *end*. Между ними заключен список компонентов, называемых *полями*, с указанием имен полей и типа каждого поля.

Структура объявления типа записи такова:

Type
<имя типа> = Record
<имя поля>: <тип компонентов>;
...
<имя поля>: <тип компонентов>;
End;
VAR
<имя переменной>: <имя типа>;

Идентификатор поля должен быть уникален только в пределах записи, однако лучше его сделать уникальным в пределах всей программы. Объем памяти, необходимый для записи, складывается из длин полей. Значения полей записи могут быть использованы в выражениях. *Обращение к значению поля* осуществляется с помощью идентификатора переменной и идентификатора поля, разделенных точкой. Такая комбинация называется *составным именем*.

Обращение к полям такой записи имеет громоздкий вид, для решения этой проблемы в языке Паскаль предназначен оператор With, который имеет следующий формат:

With <переменная типа запись> do <оператор>;

Один раз, указав переменную типа запись в операторе *With*, можно работать с именами полей как с обычными переменными.

11.15 Структура и функции работы с файлами.

Большие совокупности данных, например сведения обо всех учащихся школы или телефонный справочник, удобно иметь записанными во внешней памяти в виде последовательности сигналов. Любой язык программирования должен содержать средства для организации хранения информации на внешних запоминающих устройствах и доступа к этой информации. В Паскале для этих целей предусмотрены специальные объекты - файлы.

Файлом называется совокупность данных, записанная во внешней памяти под определенным именем.

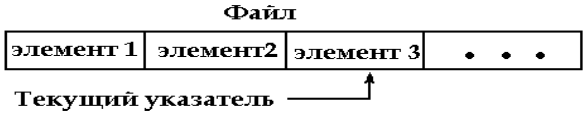
Любой файл имеет три особенности. Во-первых, у него есть *имя*, что дает возможность программе работать одновременно с несколькими файлами. Во- вторых, он содержит

компоненты одного типа. Таким компонентом может быть любой тип Паскаля, кроме файлового. Например, допускается файл записей или файл строк, но нельзя создать файл файлов. В-третьих, *длина вновь создаваемого файла* никак не оговаривается при его объявлении и *ограничивается только емкостью устройств внешней памяти*.

Для доступа к файлу описывается специальная *файловая переменная*, которая считается представителем файлов в программе.

Формат описания файлового типа

Type
<имя типа> =<тип компонентов>;
Var
<F > : file of <имя типа>;



Файл можно представить как *список значений одного и того же (базового) типа*. Все элементы файла считаются пронумерованными, начальный элемент имеет нулевой номер.

В любой момент времени программе доступен только один элемент файла, на который ссылается *текущий указатель* (указатель обработки). Часто позицию размещения доступного элемента называют *текущей позицией.*

Как правило, все действия с файлом (чтение из фала, запись в файл) производятся поэлементно, причем в этих действиях участвует тот элемент файла, который обозначается текущим указателем. В результате совершения операций текущий указатель может перемещаться.

По способу доступа к элементам различают фалы последовательного и прямого доступа. *Файлом последовательного доступа* называется файл, к элементам которого обеспечивается доступ в такой же последовательности, в какой они записывались. *Фалом прямого доступа* называется файл, доступ к элементам которого осуществляется по адресу элемента. Например, для поиска нужного элемента в последовательном файле необходимо, начиная с нулевого, перемещать указатель обработки до тех пор, пока он не будет указывать на искомый элемент, а при поиске нужного элемента в файле прямого доступа достаточно указать номер его позиции.

При организации данных в файл последовательного доступа *нельзя одновременно* читать данные из файла и записывать данные в файл, т.к. для чтения некоторого элемента последовательного файла указатель обработки помещен на данный элемент, а для записи нового элемента этот указатель одновременно должен быть в конце файла.

В Паскале имеется 3 класса файлов:

- Текстовые
- Типизированные
- Нетипизированные

Средства обработки файлов

Каждому файлу в языке ставится в соответствие файловая переменная, которая должна быть связана с конкретным внешним файлом. Поэтому перед началом работы с файлом необходимо установить данное соответствие. Для этого в языке используется процедура:

Assign(Var F; Name:String);

где *F*- переменная любого файлового типа, а строковое выражение *Name* содержит полное имя файла, удовлетворяющее требованиям операционной системы.

Процедура *Assign* всегда предшествует другим процедурам работы с файлами, так как ставит в соответствие конкретному файлу на внешнем устройстве логическую файловую переменную языка, к которой впоследствии будут обращаться все другие файловые

процедуры. Недопустимо использование процедуры *Assign* для уже открытого файла. Это значит, что если было назначено имя конкретного набора данных файловой переменной с помощью процедуры *Assign*, а затем этот файл был открыт, то, прежде чем использовать ту же файловую переменную для нового набора данных, необходимо с помощью процедуры *Close* закрыть этот файл.

Для работы с файлом прежде всего необходимо его открыть. В языке Паскаль предусмотрены для этого две процедуры:

Reset(var F : file); - открывает существующий файл;

Rewrite(var F : file); -создает и открывает новый файл.

При описании обеих процедур параметр *file* означает файловую переменную любого типа. Открытие внешнего файла с помощью процедуры *Reset* в случае его отсутствия на диске может привести к ошибке при выполнении программы.

Пример1. Стандартное открытие файла

```
Assign(F, ' ') ;
Reset(F) ;
```

Процедура *Rewrite* создает и открывает новый файл. Использование этой процедуры требует особого внимания. При попытке создать и открыть новый файл с именем уже существующего на диске набора данных действие процедуры *Rewrite* сведется к удалению этого набора и созданию нового пустого файла с тем же именем.

Если процедура *Rewrite* используется для текстового файла, то к открываемому новому набору данных в дальнейшем могут быть применимы *только операции записи*.

Операция закрытия файла является логическим окончанием работы с любым открытым файлом. Для этого служит процедура

Close (Var F);

Использование процедуры *Close* позволяет устранить связь файловой переменной с внешним файлом, установленную с помощью процедуры *Assign.*

К языковым средствам обслуживания файлов необходимо отнести процедуры переименования и удаления неоткрытых файлов. Использование этих процедур не зависит от типа файла.

Rename(var F; NewName : string);

Процедура переименовывает неоткрытый файл *F* любого типа. Новое имя задается строкой *NewName.*]

Erase(var F) ;

Процедура удаляет неоткрытый внешний файл любого типа, задаваемый переменной *F*.

Обе процедуры *нельзя использовать* для уже *открытых* файлов. Операции удаления и переименования осуществляются *только для реально существующих файлов*, иначе возникает ошибка выполнения программы.

Для того чтобы файловые операции выполнялись без ошибок, необходимо использовать специальную функцию *IOresult*. Функция работает без параметров и возвращает значение типа *integer*, представляющее статус последней выполненной операции ввода-вывода. Использование этой функции в программах возможно лишь в том случае, если на время выполнения файловых операций отключена стандартная проверка операций ввода-вывода. Для этих целей используется либо специальная опция в интегрированной системе, либо директива компилятора *{$I},* которая может задаваться внутри текста программы.

зуются следующие процедуры и функции:

Процедуры и функции для работы с любыми файлами	Описание
ASSIGN(F,Name)	*Связь файловой переменной F с внешним*

	файлом Name. Name-переменная или константа типа String,Char. Имя файла должно быть написано в соответствии с правилами DOS. F- переменная любого файлового типа.
RESET(F[,SIZE])	*Открытие существующего файла.* Открывается существующий файл, с которым связана файловая переменная F и указатель текущего компонента файла перемещается на начало файла. Необязательный параметр SIZE используется только в нетипизированных файлах и задает размер пересылаемого элемента информации в байтах.
REWRITE(F[,SIZE])	*Открытие нового пустого файла.* Открывается новый пустой файл, и ему присваивается имя, заданное процедурой ASSIGN. Если файл с таким именем уже существует, то он уничтожается. Необязательный параметр SIZE используется только в нетипизированных файлах и задает размер пересылаемого элемента информации в байтах.
ERASE(F)	*Уничтожение файла.* Удаляет неоткрытый внешний файл, задаваемый переменной F.
RENAME (F,NewName)	*Переименование файла.* Переименовывает неоткрытый внешний файл, задаваемый переменной F, новое имя задается строкой NewName.
CLOSE(F)	*Закрытие файла.*
EOF(F)	*Конец файла.*

Текстовые файлы

Текстовые файлы представляют собой совокупность символов, разделенных на строки, причем в конце каждой строки стоит признак конца строки. Особенностью работы с текстовыми файлами является то, что параметры, значения которых вводятся и выводятся с помощью процедур *READ* или *WRITE*, могут быть не только типа *String* или *Char*, но и других типов (целых, вещественных - при вводе, целых, вещественных, логических - при выводе).

При работе с текстовыми файлами необходимо, прежде всего, объявить соответствующую файловую переменную:

```
Var
  F:text;
```

Начало работы с текстовым файлом - стандартное: файловой переменной ставится в соответствие имя файла (процедура *Assign*), а затем открывается новый текстовый файл (процедура *Rewrite*) или открывается существующий текстовый файл (процедура *Reset*).

Далее в таблице представлены процедуры и функции для работы с текстовыми файлами.

Процедуры и функции для работы с текстовыми	Описание

файлами	
APPEND(F)	*Открытие существующего файла.* Открывается существующий файл, с которым связана файловая переменная F и указатель текущего компонента файла перемещается на конец файла. F- переменная любого файлового типа. После открытия текстовый файл можно дополнять информацией, начиная с конца строки.
READ(F,<список>)	*Чтение из файла.*
READLN(F,<список>)	*Чтение строки из файла.*
SETTEXTBUF(F,BUF[,SIZE])	*Назначение буфера ввода- вывода.* Для файла, с которым связана файловая переменная F, назначается буфер ввода- вывода в виде переменной BUF любого типа.
WRITE(F,<список>)	*Запись в файл.*
WRITELN(F,<список>)	*Запись строки в файл.*
EOLN(F)	*Коней строки файла.*
EOF(F)	*Конец файла.*

Типизированные файлы

К типизированным файлам относятся файлы *строго определенного типа.* Чаще всего это файлы, состоящие из записей. Они применяются для создания различных баз данных. Стандартное задание в программе такой файловой переменной осуществляется следующим образом:

```
Type
FileRec = record
                    ...
end;
Var
F : file of FileRec;
```

Если в текстовых файлах содержимое рассматривается как наборы символов, подготовленные специальным образом с учетом общепринятых соглашений о представлении текстовой информации, то в типизированных файлах их содержимое рассматривается как *последовательность записей определенной структуры*. Единицей измерения такого набора данных является сама запись. Длина записи определяется как *SizeOf(FileRec).* Так как длина любого компонента типизированного файла строго постоянна, это дает возможность организовать прямой доступ к любому компоненту по его порядковому номеру, поэтому типизированные файлы часто называют *файлами прямого доступа*.

Seek(var F; NumRec: Longint)

Процедура устанавливает текущую файловую переменную *F* на запись с номером *NumRec; F* - файловая переменная для типизированных и нетипизированных наборов данных. При открытии типизированного файла текущая позиция для работы с ним установлена на начало первой записи, которая по принятым соглашениям имеет номер 0, т. е. номер физической записи на единицу меньше номера логической записи.

Средства работы с типизированными файлами

Для работы с файлами прямого доступа дополнительно можно использовать следующие средства:

Truncate(var F);

Процедура уничтожает все компоненты файла F, начиная с места текущего положения файлового указателя.

FilePos(var F) : Longint

Функция возвращает для файла *F* текущую файловую позицию (номер записи, на которую она установлена) в виде значения типа *Longint*

FileSize(var F) : Longint

и его размер (*количество записей*) в виде значения типа *Longint.*

Для пустого файла вызов *FileSize* возвращает значение 0. Локализация ошибок при обращении к внешним носителям для обоих функций производится через *IOresult.*

Для того чтобы очередная запись могла быть записана в конец типизированного файла, необходимо перевести текущую файловую позицию в конец файла. Когда создается новый файл, это происходит автоматически после формирования каждой очередной записи. Если файл уже создан и файловая позиция, установленная по *Seek*, находится где-нибудь в начале файла *(FilePos(F) < FileSize(F)),* то в конец файла ее позволяет перевести вызов

Seek(F, FiieSize(F));

В каждом файле число логических и физических записей совпадает, а при позиционировании номер физической записи на единицу меньше номера логической записи.

Когда записи располагаются в неотсортированном по фамилиям порядке, поиск необходимо осуществлять последовательно по всему файлу. Такое расположение записей требует значительных расходов ресурсов системы для поиска нужной записи. Любая программная система по манипулированию базами данных всегда имеет в своем составе средства *упорядочения записей по ключу.*

Ключом называется совокупность знаков, идентифицирующая запись в файле. Ключ сортировки — одно или несколько полей в записи файла, по содержимому которых осуществляется упорядочение его записей, например список учащихся в классном журнале отсортирован по ключу фамилия.

Учебно-методическое обеспечение практических занятий
по дисциплине

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
(НАИМЕНОВАНИЕ ДИСЦИПЛИНЫ)

Задание 1. Форматирование текста

Создать прайс-лист с учетом представленного форматирования документа. Задание выполняется по вариантам (полный комплект материалов у преподавателя). Примеры прайс-листов приведены ниже.

Куда пойти учиться?

Почвоведение – интересная и востребованная специализация

Земля – основа жизни!

Наши выпускники проводят исследования
в Антарктиде в Мексике
в России в Европе
в Африке в Японии

почвенный глобус

БЮДЖЕТНАЯ И ПЛАТНАЯ ФОРМЫ ОБУЧЕНИЯ

Кафедра Почвоведения и экологии почв Биолого-почвенного факультета Санкт-Петербургского государственного университета приглашает выпускников школ на образовательную программу «Почвоведение»
Подготовка бакалавров ведется в рамках профиля «Экология и биология почв, землепользование»

WWW.SOIL.PU.RU

Магистры почвоведения

Наши выпускники востребованы:

- в Федеральной службе земельного кадастра России
- в государственных органах экологического контроля и управления земельными ресурсами
- в экологических и контролирующих службах предприятий и организаций (экологический аудит и менеджмент)
- в сфере образования (ВУЗы, колледжи, школы)
- в сельскохозяйственном секторе
- в научных и проектно-изыскательских институтах, бюро, фирмах и других организациях, использующих знания и информацию о почвах (экология, землеустройство, ландшафтный дизайн, агрономия, биология, картография, строительство, криминалистика, медицина, археология) и др.

Кафедра Почвоведения и экологии почв
Биолого-почвенного факультета СПбГУ
Санкт-Петербург, В.О., 16 линия, д. 29
(м. Василеостровская)
тел.: 321-33-62
сайт http://soil.pu.ru
e-mail: soil@bio.pu.ru

Задание 2. Работа с таблицами.

1. Создайте в новом документе таблицу с заданным форматированием (см. Таблица 1).

Таблица 1

Название страны	Население млн. чел.		Плотность чел. на кв. км.		Площадь, млн. кв. км.
	1970 г	1989 г	1970 г	1989 г	
Австралия и Океания	19	26	2	3	8,5
Африка	361	628	12	21	30,3
Европа	642	701	61	67	10,5
Южная Америка	190	291	11	16	17,8
Северная и Центральная Америка	320	422	13	17	24,3
Азия	2161	3133	49	71	44,4
Весь мир	3693	5201	27	38	135,8

2. Выполните сортировку в таблице, упорядочив строки по площади страны в порядке возрастания.

3. Создайте таблицу и вычислите выражения согласно изложенному ниже алгоритму. В этом примере необходимо суммировать значения в таблице.

Таблица 2

1. A	*B*	*C*	*D*	*E*	*F*	*G*
2.	2 квартал	Восток	Запад	Север	Юг	Итоги по месяцам
3.	Апрель	3524	3542	3452	3425	13943
4.	Май	3245	4254	2543	2534	12576
5.	Июнь	2435	2453	2345	2354	9587
6.	Итоги по регионам	9204	10249	8340	8313	36106

4. Создайте таблицу и выполните форматирование по образцу Таблица 3

Таблица 3.

<table>
<tr><td colspan="4">Технические характеристики</td><td colspan="2">Информация для заказа</td></tr>
<tr><td rowspan="11">Печать</td><td colspan="3">Технология HP PhotoRet II с многослойным наложением цвета 600×600: черная с технологией улучшения разрешения HP (KYt) 600×600; цветная с технологией HP PhotoRet II</td><td colspan="2">Принтер/сканер/копир типа «все в одном»</td></tr>
<tr><td colspan="3"></td><td>C6693A</td><td>HP OffisJet R65</td></tr>
<tr><td colspan="3"></td><td>C6692A</td><td>HP OffisJet R45</td></tr>
<tr><td>Метод печати</td><td colspan="2">Термальная по запросу струйная печать</td><td colspan="2">Кабели</td></tr>
<tr><td>Язык управления принтером</td><td colspan="2">P PC Level 3 или PCL3 GUI</td><td>C2946A</td><td>Параллельный кабель IEEE 1235A-C, 3 м</td></tr>
<tr><td>Нагрузка</td><td colspan="2">3000 страниц в месяц (в среднем)</td><td>C2947A</td><td>Параллельный кабель IEEE 1235A-C, 10 м</td></tr>
<tr><td>Скорость печати (с/мин)</td><td>Черная</td><td>Цветная</td><td colspan="2">Струйные принтерные картриджи</td></tr>
<tr><td>Быстрая
Обычная
Наилучшая</td><td>11
5,1
4,4</td><td>8,5
3,6
1</td><td>51645A</td><td>Большой черный картридж HP</td></tr>
<tr><td></td><td></td><td></td><td>C1876G</td><td>Цветной картридж</td></tr>
<tr><td></td><td></td><td></td><td>C1879D</td><td>Большой трехцветный цветной картридж HP</td></tr>
<tr><td></td><td></td><td></td><td>54389G</td><td>Черный картридж</td></tr>
<tr><td>Разрешение принтера</td><td>Черная</td><td>Цветная</td><td colspan="2" rowspan="3"></td></tr>
<tr><td>Быстрое
Обычное
Наилучшее</td><td>600×300
600×300
600×600</td><td>300×300
600×600
600×600</td></tr>
<tr><td>Встроенные шрифты</td><td colspan="2">Courier, Courier Italic; CG Times, CG Times Italic; Letter Gothic</td></tr>
</table>

Задание 3. Работа с редактором формул.

Необходимо в редакторе формул набрать представленные математические формулы.

а) $\lim_{x\to -2}\frac{x^2+6x+8}{x^3+8}$

б) $f(x)=4x^5-3sinx+5ctgx$

в) $\int_0^4 3\sqrt[3]{x^2}dx$

г) $f(x)=\frac{x^2}{x^3+4}$

д) $f(x)=cosx-\frac{1}{3}\cos^3 x$

е) $\int_0^4 3\sqrt{x}dx.$

ж) $f(x)=\frac{x^2-1}{x^2+1}$

з) $f(x)=sinx+\frac{1}{4}\sin^3 x.$

и) $\int_0^2(3x^2-1)dx$

$$\int\frac{dx}{\sqrt{a^2-x^2}}=\arcsin\frac{x}{a}+C=-\arccos\frac{x}{a}+C_1$$

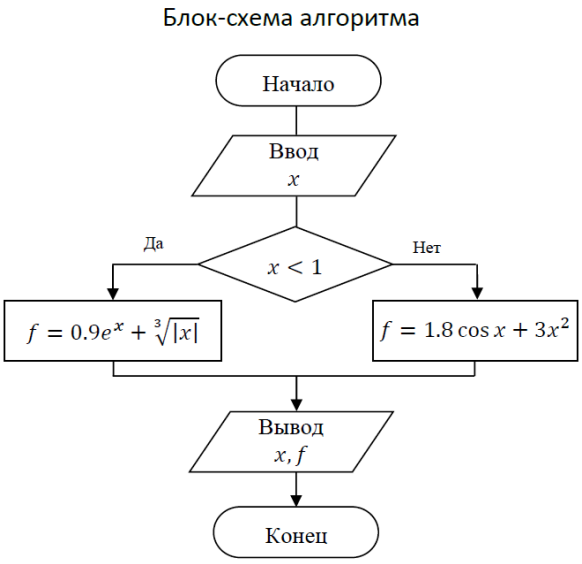
$$\sum_{n=1}^{\infty}\frac{n(2i-1)^n}{3^n}$$

$$\lim_{x\to\infty}\frac{\sqrt{x}}{\sqrt{x+\sqrt{x+\sqrt{x}}}}$$

$$y=\frac{(\alpha\sin\beta x-\beta\cos\beta x)e^{\alpha x}}{\alpha^2+\beta^2}$$

Задание 4. Работа с графическими элементами

С применением графических средств отобразить следующую блок-схему.



Задание 5 Работа в пакетом MS EXCEL

1. Создать сводную таблицу для формирования выходного документа "Штатная численность сотрудников предприятия (фирмы)" по форме:

Табельный номер	Фамилия И.О.	Должность	Должностной оклад (руб.)

Общий итог			

Содержание таблицы должно автоматически формироваться на основании содержания основной таблицы и должно быть отсортировано в порядке возрастания табельного номера.

Отобразить в виде круговой диаграммы распределение должностных окладов сотрудников.

2. Создать сводную таблицу для формирования выходного документа "Расчет фонда заработной платы предприятия (фирмы)" по форме:

Содержание таблицы должно автоматически формироваться на основании содержания основной таблицы и должно быть отсортировано в порядке убывания должностных окладов.

№ п/п	Фамилия И.О.	Должностной оклад (руб.)	Премиальные начисления (руб.)	Итого начислено (руб)
—				
	Общий итог			

Отобразить в виде гистограммы распределение должностных окладов и премиальных начислений для всех сотрудников.

3. Создать сводную таблицу для формирования выходного документа "Расчет сумм, подлежащих налогообложению для предприятия (фирмы)" по форме:

№ п/п	Фамилия И.О.	Должностной оклад (руб.)	Премиальные начисления (руб.)	Сумма, подлежащая налогообложению (руб.)
—				
	Общий итог			

Содержимое таблицы должно автоматически формироваться на основании содержания основной таблицы и должно быть отсортировало в порядке убывания сумм премиальных начислений.

Отобразить в виде линейчатой диаграммы распределение должностных окладов и премиальных начислений сотрудников.

4. Создать сводную таблицу для формирования выходного документа "Расчет сумм, подлежащих налогообложению для предприятия (фирмы)" по форме:

№ п/п	Фамилия И.О.	Должностной оклад (руб.)	Премиальные начисления (руб.)	Сумма, подлежащая налогообложению (руб.)
—				
	Общий итог			

Содержимое таблицы должно автоматически формироваться на основании содержания основной таблицы и должно быть отсортировало в порядке убывания сумм премиальных начислений.

Отобразить в виде линейчатой диаграммы распределение должностных окладов и премиальных начислений сотрудников.

5. По результатам расчетов, представленных в таблицу, построить график функции y=f(x), добавить линию тренда и представить уравнение линии тренда (все это выполняется через свойства при построении диаграммы).

X	0	1	2	3	4	5	6
У	1	5	17	37	65	101	145

Более полный комплект практических заданий по представленным разделам находится у преподавателя.

Учебно-методическое обеспечение лабораторных занятий по дисциплине

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

(НАИМЕНОВАНИЕ ДИСЦИПЛИНЫ)

Лабораторная работа №1. Линейные программы.

Цель: *Освоить работу в среде Turbo Pascal и научиться писать самые простые программы – линейные.*

1. Приступая к написанию программы, четко определите, что является ее исходными данными и что требуется получить в результате.
2. Выбирайте тип переменных с учетом диапазона и требуемой точности представления данных.
3. Давайте переменным имена, отражающие их назначение.
4. Ввод с клавиатуры предваряйте приглашением, а выводимые значения – пояснениями. Для контроля сразу же после ввода выводите исходные данные на дисплей.
5. До запуска программы подготовьте тестовые примеры, содержащие исходные данные и ожидаемые результаты. Отдельно проверьте реакцию программы на неверные исходные данные.
6. При записи выражений обращайте внимание на приоритет операций.
7. Разделяйте данные при вводе пробелами, символами перевода строки или табуляции.
8. Тщательно форматируйте текст программы и снабжайте его содержательными комментариями.

Пример программы расчета по заданной формуле.

```
program formula;
var
  a,x,y:real;                          {Исходные данные и результат}
  temp:real;                           {Промежуточная переменная}
begin
   writeln('Введите a и x:');
   readln(a,x);
   writeln('Исходные данные:');        {Контрольный вывод исходных данных}
   writeln(a= ',a:6:2.'   x=',x:6:2);
   temp:=2*sin(2*a)/cos(2*a);
   y:=(sqrt(pi*x)-exp(0.2*sqrt(a))+temp+1.6*ln(sqr(x))/ln(10))/
     (temp*1/cos(x));
   writeln('Результат:  y=',y:6:2);
   readln;
end.
```

Индивидуальные задания

Написать программу для расчета по двум формулам. Предварительно подготовьте тестовые примеры для второй формулы с помощью калькулятора (результаты вычисления по обеим формулам должны совпадать).

Вариант 1

$z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha);$

$z_2 = \frac{1}{4} - \frac{1}{4}\sin(\frac{5}{2}\pi - 8\alpha).$

Вариант 2

$z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha;$

$z_2 = 2\sqrt{2}\cos\alpha \cdot \sin\left(\frac{\pi}{4} + 2\alpha\right).$

Вариант 3

$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha};$

$z_2 = 2\sin\alpha.$

Вариант 4

$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha};$

$z_2 = tg3\alpha.$

Вариант 5

$z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha;$

$z_2 = \cos^2\alpha + \cos^4\alpha.$

Вариант 6

$z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha;$

$z_2 = 4\cos\frac{\alpha}{2} \cdot \cos\frac{5}{2}\alpha \cdot \cos 4\alpha.$

Вариант 7

$z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right);$

$z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}.$

Вариант 8

$z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1;$

$z_2 = \sin(y + x) \cdot \sin(y - x).$

Вариант 9

$z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2;$

$z_2 = -4\sin^2\frac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta).$

Вариант 10

$$z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)};$$

$$z_2 = ctg\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right).$$

Вариант 11

$$z_1 = \frac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha};$$

$$z_2 = \frac{1 - tg\alpha}{1 + tg\alpha}.$$

Вариант 12

$$z_1 \frac{\sin 4\alpha}{1 + \cos 4\alpha} \frac{\cos 2\alpha}{1 + \cos 2\alpha};$$

$$z_2 = ctg\left(\frac{3}{2}\pi - \alpha\right).$$

Вариант 13

$$z_1 = \frac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)};$$

$$z_2 = \frac{1 + \sin 2\beta}{\cos 2\beta}.$$

Вариант 14

$$z_1 = \frac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha};$$

$$z_2 = tg\,2\alpha + \sec 2\alpha.$$

Вариант 15

$$z_1 \frac{\sqrt{2b + 2\sqrt{b^2 - 4}}}{\sqrt{b^2 - 4} + b + 2};$$

$$z_2 = \frac{1}{\sqrt{b + 2}}.$$

Лабораторная работа №2. Разветвляющиеся вычислительные процессы. Условный оператор IF.

Цель: *Научиться применять оператор ветвления IF при составлении программ.*

Формат оператора:

If выражение then оператор_1 [else оператор_2]

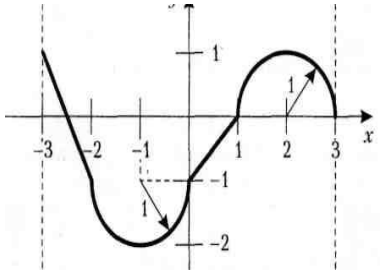
Сначала вычисляется выражение, которое должно иметь логический тип. Как правило, в выражении используются знаки *операций отношения* (<, >, =,<>, <=, >=). Если требуется проверить несколько условий, их объединяют знаками *логических операций* and (и),

or(или), xor (исключающее ИЛИ) и not (отрицание).

Если выражение имеет значение true, выполняется первый оператор, иначе – второй. Ветвь else может отсутствовать. После выполнения операторов из соответствующей ветви управление передается оператору, следующему за условным.

Пример.

Пример программы, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика на интервале [-3;3].



Запишем функцию в виде формул:

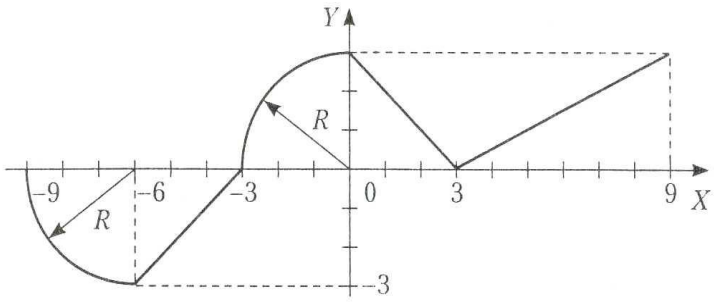
$$y=\begin{cases} -2x-5, -3\le x<-2 \\ -\sqrt{1-(x+1)^2}-1, -2\le x<0 \\ x-1, 0\le x<1 \\ \sqrt{1-(x-2)^2}, 1\le x<3 \end{cases}$$

```
program calc_fun;
var x,y:real;
begin
    writeln('Введите значение аргумента:');
    readln(x);
    if (x<-3) or (x>3) then begin
      writeln('Значение должно принадлежать интервалу [-3;3]');
      exit
    end;
    if x<-2 then y:=-2*x-5
    else if x<0 then y:=-sqrt(1-sqr(x+1))-1
    else if x<1 then y:=x-1
    else y:=sqrt(1-sqr(x-2));
    writeln('Для  x=',x:6:2, '  значение функции   y=',y:6:2);
readln;
end.
```

Индивидуальные задания

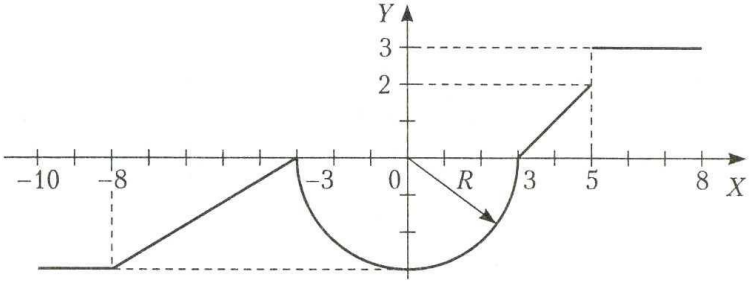
Вариант 1

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



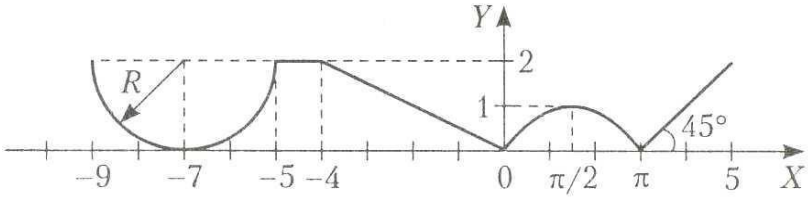
Вариант 2

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



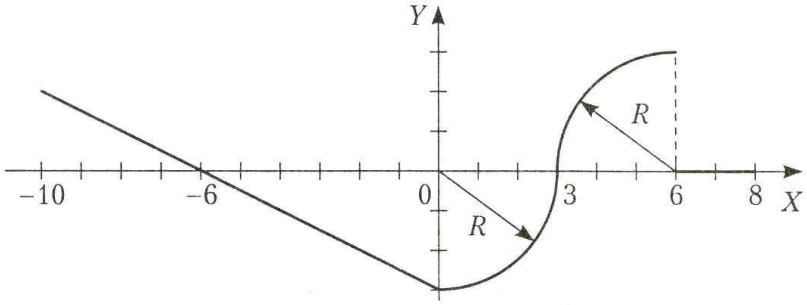
Вариант 3

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



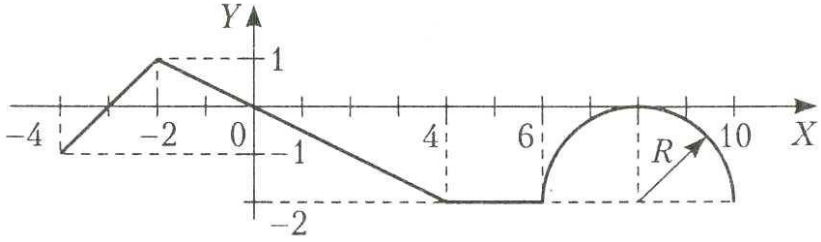
Вариант 4

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



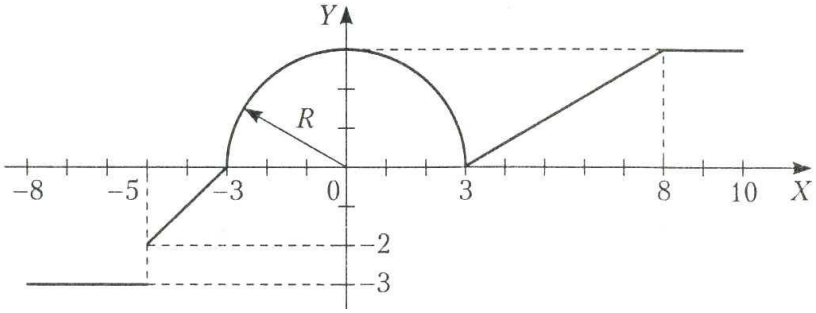
Вариант 5

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



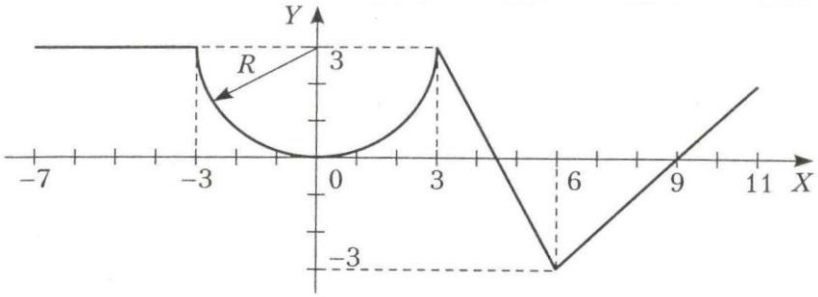
Вариант 5

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



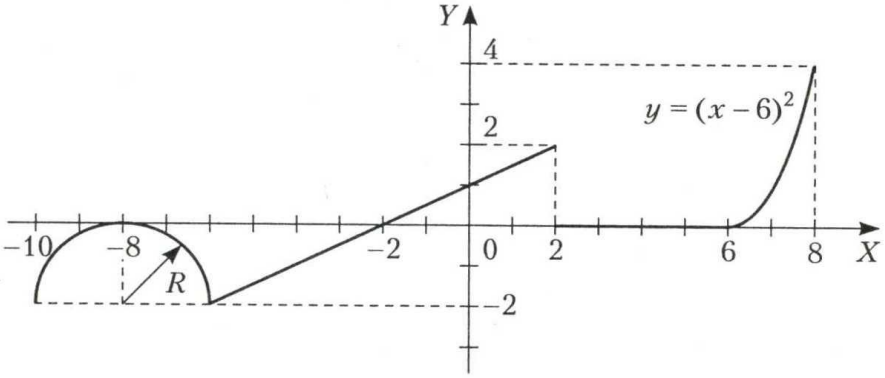
Вариант 6

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



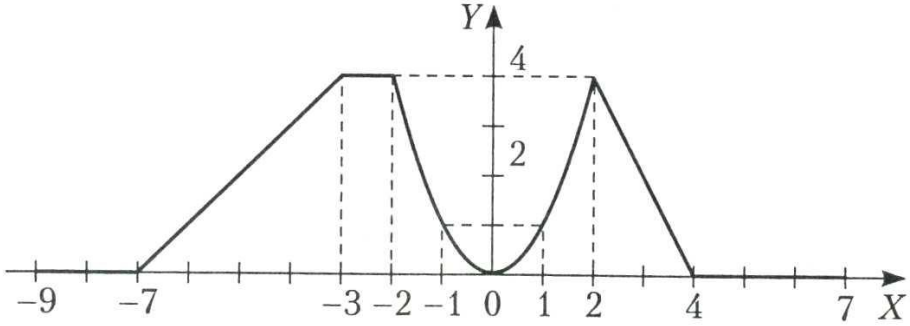
Вариант 7

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



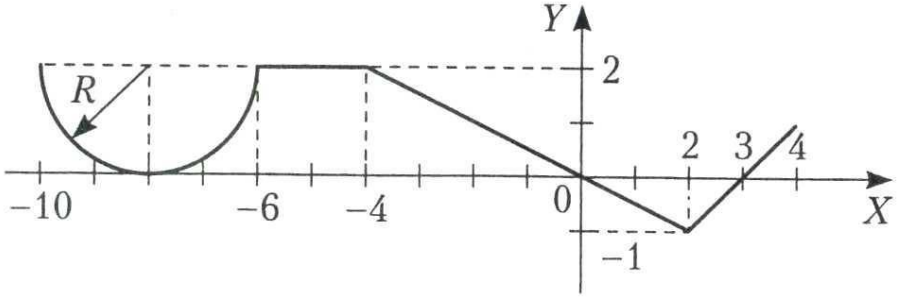
Вариант 8

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



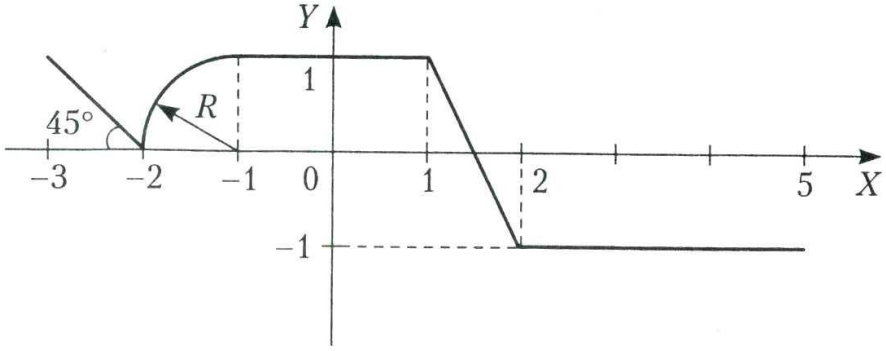
Вариант 9

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



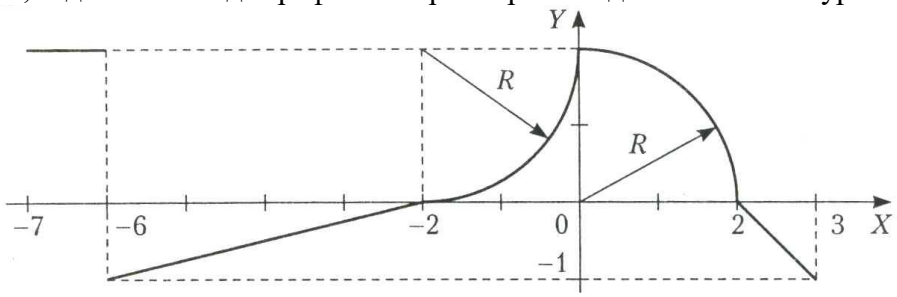
Вариант 10

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



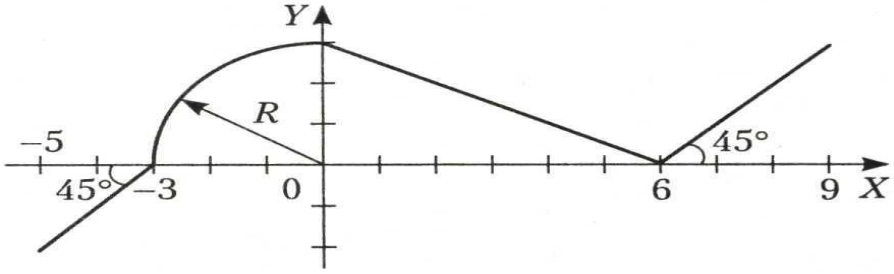
Вариант 11

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



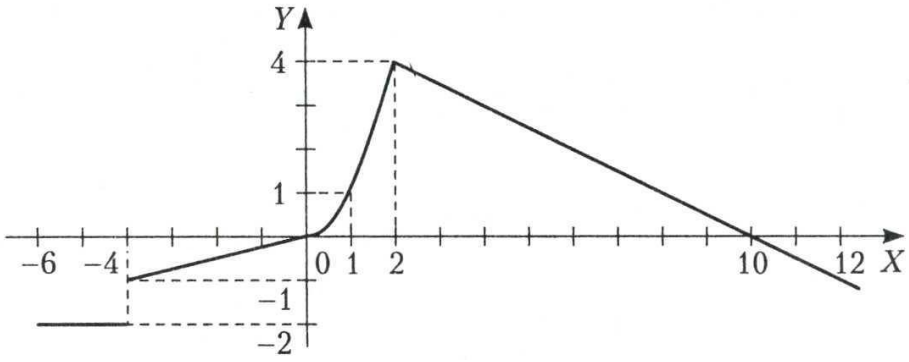
Вариант 12

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



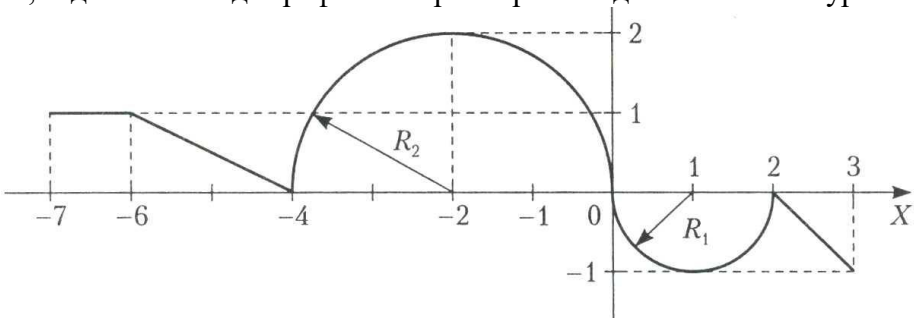
Вариант 13

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



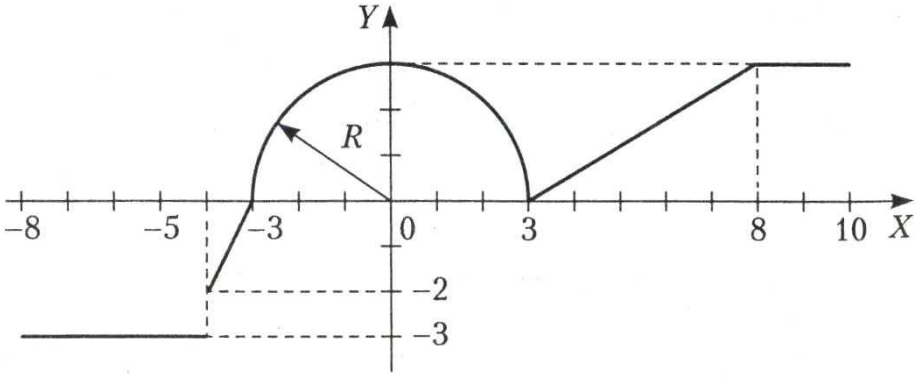
Вариант 14

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



Вариант 15

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.



Лабораторная работа №3. Организация циклов.

Цель: *получить навыки циклических вычислений с применением различных операторов циклов*

Цикл – это фрагмент программы, повторяемый многократно. В Паскале три оператора цикла – *while, repeat и for*. Все циклы имеют схожую структуру (Рис. 1)

Рис. 1

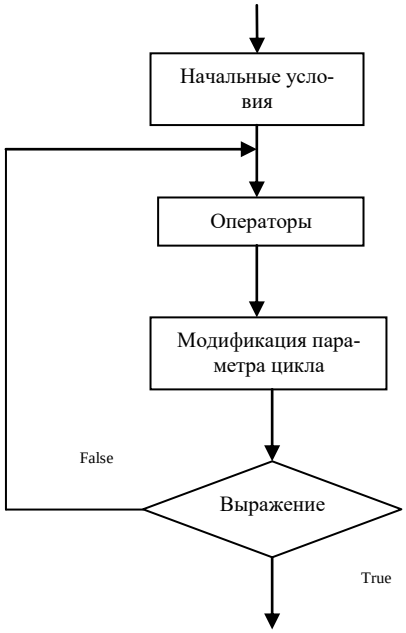


Рис. 2

Операторы, ради многократного выполнения которых организуется цикл, называются *телом цикла*. Остальные операторы служат для управления процессом повторения вычислений: это начальные установки, проверка условия продолжения цикла и модификация параметра цикла. Один проход цикла называется *итерацией*. На этапе начальных установок (до входа в цикл) задаются значения переменных, которые в нем используются. Эти значения могут задаваться явно или неявно. Цикл завершается, если условие его продолжения не выполняется. Возможно принудительное завершение как текущей итерации (для этого применяется процедура continue) , так и цикла в целом (процедура break и оператор goto). Передавать управление извне внутрь цикла не рекомендуется, потому что при этом не выполнятся начальные установки. Иными словами, выйти из цикла можно в любой момент, а войти – только в начало.

Цикл с предусловием while

В цикле с предусловием проверка условия продолжения цикла выполняется перед телом цикла

(Рис.2) Если при входе в цикл условие не выполняется, он не будет выполнен ни разу. Оператор цикла имеет вид

While выражение do оператор

Выражение должно быть логического типа. Если результат вычисления выражения равен *true*, выполняется расположенный после служебного слова *do* простой или составной оператор. Эти действия повторяются до того момента, пока результатом выражения не станет значение *false*. После окончания цикла управление передается следующему за ним оператору.

Пример.

Написать программу печати таблицы значений функции sin для аргумента, изменяющегося в заданных пределах с заданным шагом.

Исходными данными являются начальное значение аргумента X_n, конечное значение аргумента X_k и шаг изменения аргумента dX. Все величины – вещественные и могут принимать любые значения на числовой оси. Результатом работы программы должна быть таблица, состоящая из двух столбцов – значений аргумента и соответствующих им значений функции.

```
program tabl_fun;
var
```

```
  Xn,Xk:real;          {Начальное  и конечное значения аргумента}
  dx:real;             {Шаг изменения аргумента}
  x,y:real;            {текущие значения аргумента и функции}
begin
   writeln('Vvedite Xn,Xk,dX');      {Приглашение ко вводу данных}
   readln(Xn,Xk,dX);                 {ввод исходных данных}
writeln('_____');          {заголовок таблицы}
writeln('______X_____|_______Y______');
writeln('_____');
x:=Xn;                                {первое значение аргумента =Xn}
while x<=Xk do begin                  {заголовок цикла}
   y:=sin(x);                         {вычисление значения функции}
   writeln('|',x:7:2,'    |',y:7:2,'     |');   {вывод строки табл.}
   x:=x+dx;                           {переход к следующему значению аргумента}
        end;
   writeln('_____') ;
readln; end.
```

Выполнить программу для Xn=-4, Xk=4, dx=1.

Индивидуальные задания

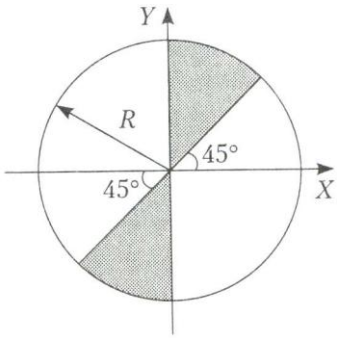
Задание 1

Вычислить и вывести на экран в виде таблицы значения функции, заданной графически (см. задание 1 лабораторной работы №2), на интервале от $x_{нач}$ до $x_{кон}$ с шагом dx. Интервал и шаг задать таким образом, чтобы проверить все ветви программы. Таблицу снабдить заголовком и шапкой.

Задание 2

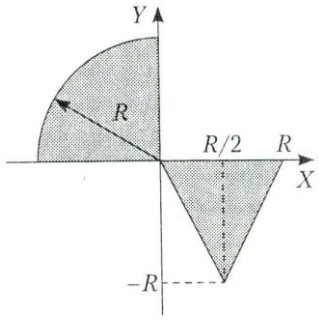
Подсчитать какое количество точек, заданных своими координатами, попадает в заштрихованную область

Вариант 1

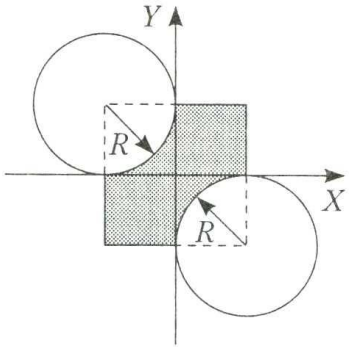


2

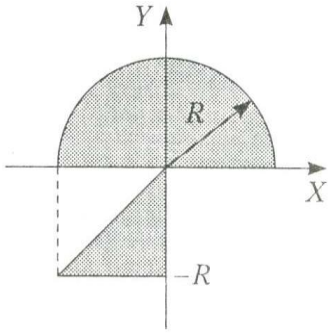
Вариант 2



Вариант 3

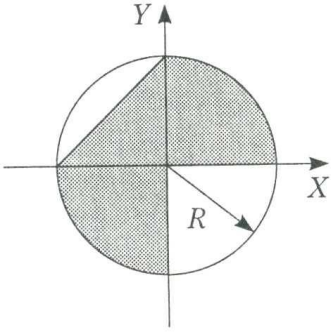


Вариант 4

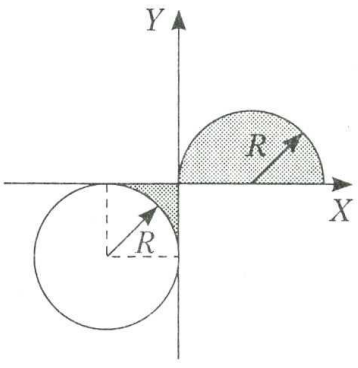


\

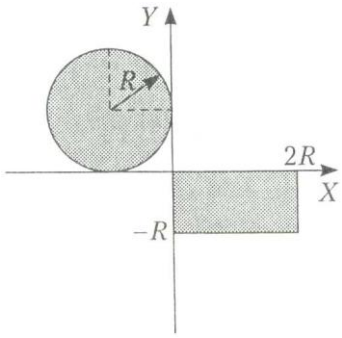
Вариант 5



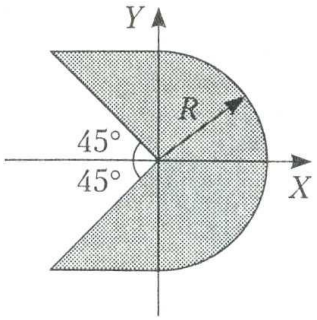
Вариант 6



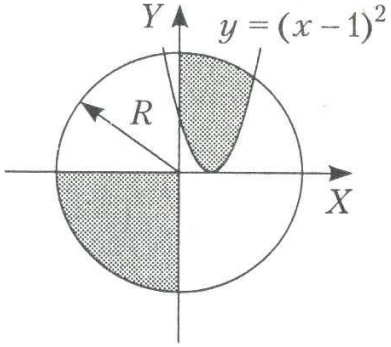
Вариант 7



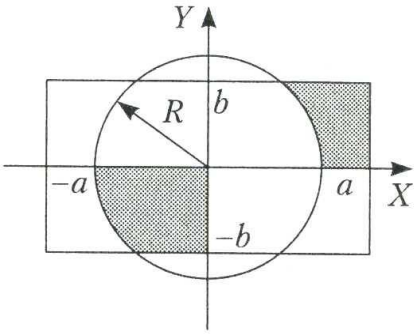
Вариант 8



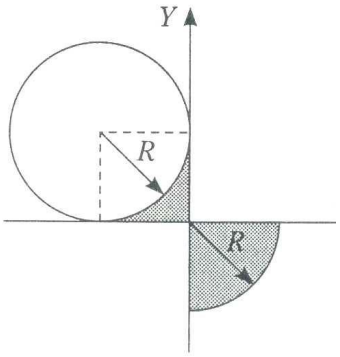
Вариант 9



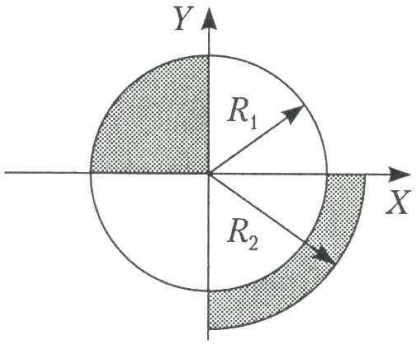
Вариант 10



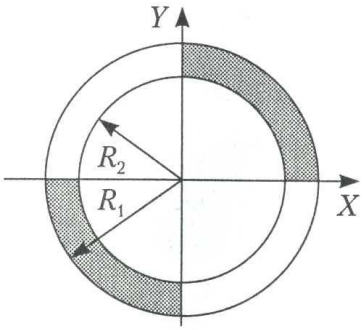
Вариант 11



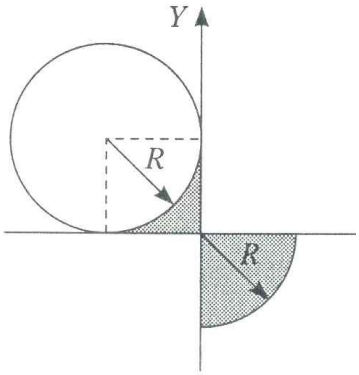
Вариант 12



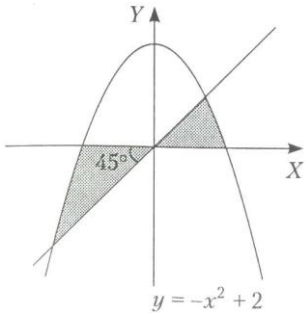
Вариант 13



Вариант 14



Вариант 15



Лабораторная работа №4. Организация вложенных циклов.

Цель: *Рассмотреть программирование задач с вложенными циклами.*

Пример.

Написать программу вычисления значения функции sin с помощью степенного ряда с точностью ε по формуле

$$y = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

Этот ряд сходится на всей числовой оси. Для достижения заданной точности требуется суммировать члены ряда до тех пор, пока абсолютная величина очередного члена не станет меньше или равна ε.

```
program sinus;
const MaxIter=500;    {максимально допустимое количество итераций}
var x,eps:double;     {аргумент и точность}
   c,y:double;        {член ряда и его сумма}
   n:integer;         {номер члена ряда}
   done:boolean;      {признак достижения точности}
begin
    writeln('Введите аргумент и точность:');
    readln(x,eps);
    done:=true;
    c:=x;                  {первый член ряда}
    y:=c;                  {начальное значение суммы}
    n:=1;
    while abs(c)>eps do begin
        c:=-c*sqr(x)/2/n/(2*n+1);     {очередной член ряда}
        y:=y+c;                       {добавление члена ряда к сумме}
        inc(n);                       {переход к следующему члену ряда}
        if n<=MaxIter then continue;
        writeln('ryd rasxoditsya');
        done:=false;break             {аварийный выход из цикла}
    end;
    if done then
      writeln('Аргумент:',x:10:6,#13#10,
           'Значение функции:',y:10:6,#13#10,
           'Вычислено с точностью',eps:8:6,'за',n,'итераций');
```

```
    readln;
end.
```

Индивидуальные задания

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от $x_{нач}$ до $x_{кон}$ с шагом dx с точностью ε. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

1. $\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots\right),\quad |x|>1.$

2. $e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots,\quad |x|<\infty.$

3. $e^{x}=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots,\quad |x|<\infty.$

4. $\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}-\ldots,\quad -1<x\le 1.$

5. $\ln\frac{1+x}{1-x}=2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n+1)}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right),\quad |x|<1.$

6. $\ln(1-x)=-\sum_{n=1}^{\infty}\frac{x^n}{n}=-\left(x+\frac{x^2}{2}+\frac{x^4}{4}+\ldots\right),\quad -1\le x<1.$

7. $arcctgx=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=\frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}+\ldots,\quad |x|\le 1.$

8. $arctgx=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\ldots,\quad x>1.$

9. $arctgx=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)}=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots,\quad |x|\le 1.$

10. $\text{arthx}=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\ldots,\quad |x|<1.$

11. $\text{arthx}=\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots, \qquad |x|>1.$

12. $arctgx=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\ldots, \qquad |x|<-1.$

13. $e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\ldots, \qquad |x|<\infty.$

14. $\cos x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{4!}+\ldots, \qquad |x|<\infty.$

15. $\frac{\sin x}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}-\ldots, \qquad |x|<\infty.$

Лабораторная работа №5. Одномерные массивы

Цель: *познакомить студентов с понятием структурированных переменных, их особенностями, ввести понятие массива, рассмотреть приемы работы с данными типа массив.*

Описание массива

Чтобы описать массив, надо сообщить компилятору:

- сколько в нем элементов;
- какого типа эти элементы;
- как они нумеруются.

Массив не является стандартным типом данных, поэтому он задается в разделе описания типов:

type *имя_типа* = array *[тип_индекса]* of *тип_элемента*

Здесь type — признак начала раздела описания типов, array и of — ключевые слова, тип индекса задается в квадратных скобках, например:

type mas = array [1 .. 10] of real;

Здесь описан тип массива из вещественных элементов, которые нумеруются от 1 до 10. После задания типа его можно использовать точно так же, как любой стандартный тип, то есть описывать с его помощью переменные, например:

var a, b : mas;

Компилятор, встретив такой оператор, выделит по 60 байт под каждый из массивов a и b (10 элементов по 6 байт). К элементу массива обращаются, указав его имя, за которым в квадратных скобках записывается порядковый номер элемента:

a[4] b[i]

С элементом массива можно делать все, что допустимо для переменных того же типа.

Тип элементов массива может быть любым, кроме файлового, тип индексов — интервальным, перечисляемым или byte. При описании типа индексов можно использовать только константы или константные выражения. Переменные не допускаются, потому что место под массив резервируется до выполнения программы.

Если тип массива используется только в одном месте программы, можно задать тип прямо при описании переменных, например:

```
const n=100:
var x,y : array [1 . . n] of integer;
```

С массивами в целом можно выполнять только одну операцию — *присваивание.* При этом массивы должны быть одного типа, например:

```
x := y;
```

Пример.

Написать программу, которая для 10 целочисленных элементов определяет, сколько положительных элементов располагается между максимальным и минимальными элементами.

Алгоритм поиска максимума:

1. Принять за максимальный первый элемент массива.
2. Просмотреть массив, начиная со второго элемента.
3. Если очередной элемент оказывается больше максимального, принять его за максимальный.

Для решения поставленной задачи нам требуется знать не значение максимума, а его положение в массиве, то есть индекс:

```
program num_positive_1;
uses crt;
const
    n=10;
    a:array[1..n] of integer=(1,3,-5,1,-2,1,-1,3,8,4);
var
  i:integer;              {индекс текущего элемента}
  imax:integer;           {индекс максимального элемента}
  imin:integer;           {индекс минимального элемента}
  ibeg:integer;           {начало интервала}
  iend:integer;           {конец интервала}
  count:integer;          {количество положительных элементов}
begin
    clrscr;
    for i:=1 to n do write(a[i]:3);writeln;       {отладочная часть}
    imax:=1;imin:=1;                    {начальные значения номеров макс. и мин. эл-ов}
    for i:=1 to n do begin
       if a[i]>a[imax] then imax:=i;    {новый номер максимума}
       if a[i]<a[imin] then imin:=i;    {новый номер минимума}
    end;
    writeln ('max=',a[imax],'   min=',a[imin]);          {отладочная печать}
    if imax<imin then ibeg:=imax else ibeg:=imin;  {левая граница}
    if imax<imin then iend:=imin else iend:=imax;  {правая граница}
    writeln('ibeg=',ibeg,'   iend=',iend);         {отладочная печать}
    count:=0;
    for i:=ibeg+1 to iend-1 do          {подсчет кол-ва положительных}
       if a[i]>0 then inc(count);
    writeln('Количество положительных: ',count);
    readln;
end.
```

Замечание: inc(x) – увеличивает на 1;

dec(x) – уменьшает на 1.

Индивидуальные задания.

1. Заменить все положительные элементы целочисленного массива, состоящего из n элементов, на значение минимального.
2. Дан массив, состоящий из n элементов. Переставить в обратном порядке элементы массива, расположенные между его минимальным и максимальным элементами.
3. Дан массив, состоящий из n элементов. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Заменить каждую серию, длина которой больше k, на один наименьший элемент массива. Если таких серий нет, то массив оставить без изменений.
4. Дан массив, состоящий из n элементов. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Преобразовать массив, увеличив первую серию наибольшей длины на один элемент.
5. Дан массив, состоящий из n элементов. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Вставить перед каждой серией минимальный элемент массива.
6. Дан массив, состоящий из n элементов. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Поменять местами наибольшую первую и k-ю серии массива. Если таких серий в массиве меньше k, то вывести массив без изменений.
7. Дан целочисленный массив, состоящий из n элементов. Удалить из массива все элементы, встречающиеся менее двух раз.
8. Дан массив, состоящий из n элементов. Преобразовать его, вставив перед каждым положительным элементом минимальный элемент.
9. Дан массив, состоящий из n элементов. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Преобразовать массив, увеличив каждую его серию на один элемент.
10. Заменить все отрицательные элементы целочисленного массива, состоящего из n элементов, на значение максимального.
11. Дан массив, состоящий из n элементов. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Удалить из массива все серии, длина которых меньше k.
12. Дан целочисленный массив, состоящий из n элементов. Удалить из массива все элементы, встречающиеся более двух раз.
13. Дан массив, состоящий из n элементов. Преобразовать его, вставив после каждого отрицательного элемента минимальный элемент.
14. Дан массив, состоящий из n элементов. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Преобразовать массив, увеличив все серии наибольшей длины на один элемент.
15. Дан массив, состоящий из n элементов. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Поменять местами наименьшую последнюю и k-ю серии массива. Если таких серий в массиве меньше k, то вывести массив без изменений.

Лабораторная работа №6. Двумерные массивы.

Цель: *получение навыков программирования с двумерными массивами.*

Элементы массива могут, в свою очередь, также быть массивами, например:

```
Const n=4;m=3;
Type mas = array[1..n] of integer;
     Mas2 = array[1..m] of mas;
```

Обычно это записывается так:

```
Type mas2 = array[1..m,1..n] of integer;
```

Здесь описан тип массива, состоящего из m массивов, каждый из которых содержит n целых чисел. Иными словами, это матрица из m строк и n столбцов, или ***двумерный массив***.

Пример.

Написать программу, которая находит минимальный по модулю элемент вещественной матрицы размером 5×8 и его координаты.

В результате выполнения этой программы должно получиться три числа: значение элемента, имеющего минимальную абсолютную величину (величина вещественного типа), а также целочисленные номера строки и столбца, на пересечении которых этот элемент расположен.

Алгоритм

1. Ввести матрицу.
2. Принять элемент, расположенный на пересечении первой строки и первого столбца, за искомый (для этого запомнить его модуль, номер строки и номер столбца).
3. Поочередно просмотреть все элементы матрицы; если модуль очередного элемента оказывается меньше сохраненного, принять этот элемент за искомый (то есть запомнить его модуль, номер строки и номер столбца).
4. Вывести результаты.

Программа:

```
program min_abs_1;
uses crt;
const m=5;n=8;
var
  a:array[1..m,1..n] of real;
  i,j:integer;              {текущие значения индексов строки и столбца }
  im,jm:integer;            {значения индексов минимального элемента}
  minmod:real;              {буфер для хранения минимального модуля}
begin clrscr;        randomize;
    for i:=1 to m do
    for j:=1 to n do a[i,j]:=random(101);       {ввод элементов массива}
    writeln;writeln('Исходный массив');
    for i:=1 to m do begin
    for j:=1 to n do write (a[i,j]:6:1);writeln end;writeln;
    minmod:=abs(a[1,1]);       {начальные установки}
    im:=1;jm:=1;
    for i:=1 to m do     {цикл перебора всех строк}
    for j:=1 to n do     {цикл просмотра строки (меняется номер столбца)}
       if abs(a[i,j])<minmod then begin
          minmod:=abs(a[i,j]);im:=i;jm:=j;     {обновление минимума}            end;
    writeln ('Минимальный по модулю элемент     : ',minmod:6:1,'       Его координаты:
',im:3,jm:3);
```

```
readln;      end.
```

Размеры массива заданы именованными константами m и n, что позволяет легко их изменять.

Для ввода матрицы требуется организовать вложенные циклы. Порядок их следования не важен, главное, чтобы он совпадал с порядком ввода исходных данных. Для приведенной последовательности циклов элементы должны вводиться по строкам.

Индивидуальные задания.

1. Дан целочисленный массив. Заменить в нем все элементы, меньшие 5 числом 111.
2. Дан целочисленный массив. Заменить все его элементы их квадратами.
3. Дан действительный массив. Напечатать индексы его отрицательных элементов.
4. Создайте двумерный целочисленный массив и найдите сумму всех его элементов, имеющих оба нечетных индекса.
5. Дан целочисленный массив. Вывести на экран элементы массива, кратные 5, и их индексы.
6. Введите с клавиатуры целочисленные элементы матрицы 3*3 и вычислите сумму элементов каждого столбца.
7. Дан целочисленный массив. Найти количество отрицательных элементов этого массива, расположенных ниже левой диагонали.
8. Дан массив. Известно, что среди его элементов два и только два равны между собой. Напечатать их индексы.
9. Даны целые числа a_1, a_2, a_3. Получить целочисленный массив, для которого $B[i,j]=a_i-3a_j$.
10. Даны действительные числа a_1, a_2, ..., a_{10}, b_1, b_2, ..., b_{20}. Получить действительный массив, для которого $C[i,j]=a_j/(1+|b_i|)$.
11. Получить целочисленный массив, для которого $a_{ij}=i+2j$.
12. Создайте матрицу 5*5, значение каждого элемента которой равно сумме номера строки и столбца, на пересечении которых он находится, и вычислите сумму элементов каждой строки.
13. Получить действительный массив, первая строка которой задается формулой $a_{1j}=2j+3$, вторая строка задается формулой $a_{2j}=j+3/(2+j)$, а каждая следующая строка есть сумма двух предыдущих.
14. Дано натуральное число n. Выяснить, сколько положительных элементов содержит массив, если $a_{ij} = sin(i+j/2)$.
15. Дан целочисленный массив. Найти среднее арифметическое каждого из столбцов.

Лабораторная работа №7: Подпрограммы.

Цель: *Решение задач в виде отдельных подпрограмм, для упрощения отладки и возможной модификации программ.*

Подпрограмма представляет собой самостоятельную программную единицу, обращение к которой осуществляется по имени.

Логически законченные части программы оформляются в виде подпрограмм. Они нужны для того, чтобы упростить структуру программы и облегчить ее отладку.

Подпрограмма — это фрагмент кода, к которому можно обратиться по имени. Она описывается один раз, а вызываться может столько раз, сколько необходимо. Одна и та же подпрограмма может обрабатывать различные данные, переданные ей в качестве *аргументов.* В Паскале имеется два вида подпрограмм: процедуры и функции. Они определяются в соответствующих разделах описания до начала блока исполняемых операторов.

Само по себе описание не приводит к выполнению подпрограммы. Для того чтобы подпрограмма выполнилась, ее надо вызвать. *Вызов* записывается в том месте программы, где требуется получить результаты работы подпрограммы. Подпрограмма вызывается по имени, за которым следует *список аргументов* в круглых скобках. Если аргументов нет, скобки не нужны. Список аргументов при вызове как бы накладывается на список параметров, замещая его, поэтому они должны, попарно соответствовать друг другу.

Подобно тому, как можно при вычислении по одной и той же формуле получить разные результаты, подставляя разные значения аргументов, при каждом вызове подпрограммы можно передавать ей разные данные. Важно только, чтобы их количество и тип соответствовали тому, что требуется подпрограмме для работы.

Структура процедуры аналогична структуре основной программы: в ней есть заголовок, разделы описаний и операторов:

procedure имя (список параметров): { заголовок }
разделы описаний
begin
раздел операторов
end;

Пример.

Даны два массива целых чисел. Оформить ввод и вывод элементов в виде процедуры и найти максимальный элемент каждого массива с помощью функции.

```
uses crt;
type mas=array[1..35] of integer;
var a,b:mas;
   maxA,maxB,i:integer;
   procedure WWOD(var d:mas;n:byte);
          begin for i:=1  to n do
               d[i]:=random(100);
          end;
   procedure WIWOD(var d:mas;n:byte);
          begin writeln ('Massiv');
          for i:=1  to n do
             write(d[i]:5);
          end;
   function Max(var d:mas;n:byte):integer;
   var p:integer;
   begin  p:=d[1];
        for i:=2 to n do
        if d[i]>p then p:=d[i];
   max:=p;
   end;
begin clrscr;
```

```
        randomize;
        WWOD(a,20);
        WWOD(b,35);
        writeln;
        WIWOD(a,20);
        maxA:=Max(a,20);
        writeln('Maksimal A=',maxA);
        writeln;
        WIWOD(b,35);
        maxB:=Max(b,35);
        writeln('Maksimal B=',maxB);
    readln;
    end.
```

Индивидуальные задания.

1. Напишите программу, состоящую из трех процедур и основной программы. Первая процедура организует ввод двух целых чисел X и Y, вторая проверяет их сумму, третья выводи результат. Используйте эти процедуры в основной программе. Используйте X и Y как глобальные переменные.
2. Напишите программу вычисления площади поверхности и длины экватора на основе известного радиуса планет солнечной системы. Форму планет будем считать шаром. Вычисление площади поверхности и длины экватора оформите отдельными функциями.
3. Напишите программу поиска большего из четырех чисел с использованием подпрограммы поиска большего из двух чисел.
4. Даны координаты вершин многоугольника (x1, y1,x2,y2,…x10,y10). Напишите программу для вычисления его периметра (вычисление расстояния между вершинами оформить подпрограммой).
5. Напишите программу вычисления суммы: 1! + 2! + 3! + … + n!, используя функцию вычисления факториала числа k.
6. Напишите программу для вычисления числа сочетаний из N по M. Число сочетаний определяется по формуле N!/(M!*(N-M)!, где N – количество элементов перебора. Используйте подпрограмму вычисления факториала.
7. Напишите программу для определения НОД трех натуральных чисел.
8. Даны действительные числа s,t. Составить программу вычисления выражения f(t, -2s, 1.17) + f(2.2, t, s-t), где f(a,b,c) = (2a – b – sin(c)) / (5 +|c|).
9. Даны натуральные m и n (m<n). Составить программу, сокращающую дробь m/n.
10. Напишите программу вычисления суммы квадратов простых чисел, лежащих в интервале (M,N).
11. Напишите программу подсчета числа четных цифр, используемых в записи N-значного числа M.
12. Составьте программу вычисления суммы трехзначных чисел, в десятичной записи которых нет четных цифр.
13. Составьте программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр.
14. Составьте программу нахождения наименьшего натурального N значного числа X (X>=10), равного утроенному произведению своих цифр.
15. Составьте программу подсчета числа всех натуральных чисел, меньших M, квадрат суммы цифр которых равен X.

Лабораторная работа №8. Использование подпрограмм в Паскале.

Цель работы:

1) получить необходимые навыки при использовании операторов в среде Turbo Pascal;

2) научиться создавать, корректировать и выполнять простые программы с использованием подпрограмм.

Примеры.

Задача 1. Составьте программу, в которой при помощи функции находится max(2x, 3x-10,max(x^2-x, x^3-x^2-3x),max($a-x, x-a$)) при заданных с терминала x,a.

```
uses crt;
type mas=array[1..4] of real;
function Max(d:mas;n:integer):real;
        var i:integer;mx:real;
        begin  mx:=d[1]; for i:=2 to n do
        if d[i]>mx then mx:=d[i];
        max:=mx;
        end;
var x,t:integer;
   mx:real;
   a,b,c:mas;
begin clrscr;
     write('BBедите x,t');
     readln(x,t);
     a[1]:=sqr(x)-x;
     a[2]:=exp(3*ln(x))-sqr(x)-3*x;
     c[3]:=Max(a,2);
     writeln('Max(x^2-x,x^3-x^2-3x)=',c[3]:6:2);
     b[1]:=t-x;
     b[2]:=x-t;
     c[4]:=Max(b,2);
     writeln('Max(a-x,x-a)=',c[4]:6:2);
     c[1]:=2*x;
     c[2]:=3*x-10;
     mx:=Max(c,4);
     writeln('Max(2x,3x-10,max(x^2-x,x^3-x^2-3x),max(a-x,x-a))=',mx:6:2);
     readln;
end.
```

Задача 2. Составьте программу, в которой при помощи функции находится максимальный элемент для массивов A(22), P(12), T(33), затем общий максимальный элемент, затем при помощи процедуры все отрицательные элементы каждого из трех массивов замените на модуль первого элемента соответствующего массива для массивов A(22), P(12), T(33). Предусмотреть вывод на экран исходных и измененных массивов.

```
uses crt;
type mas=array[1..33] of integer;
var a,p,t,m:mas;
```

```
    i,maxi:integer;
    procedure WWOD(var d:mas;n:byte);
            begin for i:=1  to n do
                 d[i]:=random(101)-50;
            end;
    procedure WIWOD(var d:mas;n:byte);
            begin
            for i:=1  to n do
               write(d[i]:5);
            end;
    function Max(var d:mas;n:byte):integer;
    var mx:integer;
    begin  mx:=d[1];
          for i:=2 to n do
          if d[i]>mx then mx:=d[i];
    max:=mx;
    end;
    procedure zamena(var d:mas;k:byte);
    begin
    for i:=1 to k do if d[i]<0 then d[i]:=abs(d[1])
    end;
begin clrscr;
     randomize;
     WWOD(a,22);          WWOD(p,12);          WWOD(t,33);
     writeln;
     writeln ('Massiv A');
     WIWOD(a,22);
     m[1]:=Max(a,22);writeln;
     writeln('Maksimal A=',m[1]);
     writeln;
     writeln ('Massiv P');
     WIWOD(p,12);
     m[2]:=Max(p,12);writeln;
     writeln('Maksimal P=',m[2]);
     writeln;
     writeln ('Massiv T');
     WIWOD(t,33);
     m[3]:=Max(t,33);writeln;
     writeln('Maksimal T=',m[3]);
     maxi:=Max(m,3);
     writeln('Max=',maxi);writeln;
     zamena(a,22);
     writeln('Noviy A');
     WIWOD(a,22);
     zamena(p,12);writeln;
     writeln('Noviy P');
     WIWOD(p,12);
     zamena(t,33);writeln;
     writeln('Noviy T');
     WIWOD(t,33);
```

```
    readln;
end.
```

Индивидуальные задания.

1. Определить подпрограмму вычисления площади правильного n-угольника, вписанного в окружность заданного радиуса.
2. Определить подпрограмму вычисления площади кольца при известном наружном и внутреннем диаметре.
3. Определить подпрограмму вычисления площади сектора при известном наружном, внутреннем диаметре и центральном угле.
4. Определить подпрограмму вычисления расстояния между двумя точками в пространстве с координатами (x1,y1,z1) и (x2,y2,z2).
5. Дано целое число n (n<100), определяющее возраст человека (в годах). Определить подпрограмму, дающую для этого числа наименования "год", "года" или "лет": например 1 год, 23 года, 45 лет и т.д.
6. Определить подпрограмму вычисления факториала заданного цело численного выражения, например вычислить (2*k+1)!.
7. Определить подпрограмму вычисления корней квадратного уравнения ax2 + bx + c = 0.
8. Даны вещественные числа x1,y1,x2,y2,...,x8,y8. Найти периметр восьмиугольника, вершины которого имеют координаты (x1,y1),(x2,y2),..., (x8,y8). (Определить подпрограмму вычисления расстояния между двумя точками, заданными своими координатами.)
9. Даны целое число n и вещественные числа x1,y1,x2,y2,...,xn,yn. Найти площадь n-угольника, вершины которого имеют соответственно координаты(x1,y1),(x2,y2),...,(xn,yn). (Определить подпрограмму вычисления площади треугольника по заданными координатам вершин.)
10. Дано целое число n. Выяснить, имеются ли среди чисел n, n +1, ...,2 n близнецы, т.е. простые числа, разность между которыми равна двум. (Определить процедуру, позволяющую распознавать простые числа.)
11. Составить процедуру вычисления значения целого числа по заданной строке символов, являющихся записью этого числа в десятичной системе счисления.
12. Составить процедуру вычисления значения целого числа по заданной строке символов, являющихся записью этого числа в естнадцатеричной системе счисления (шестнадцатеричные цифры - это цифры от 0 до 9 и буквы от A до F).
13. Определить функцию, которая подсчитывает количество символов в текстовом файле (исключая цифры).
14. Определить процедуру замены в текстовом файле одного заданного символа на другой.
15. Составьте программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр.

Лабораторная работа №9. Строки.

Цель: *Освоить на практике новый тип данных – строка, научиться работать с текстом.*

1. Определение. Строки – это структурируемый тип данных, представляющий собой последовательность из набора символов кодовой таблицы. Всего 255 символов.

В отличие от массива первый (т.е. нулевой) байт памяти содержит информацию по количеству символов в строке. Поэтому, если строка описана длиной 10 символов, то любые символы введенные с терминала или добавляемые к строке после 10 будут игнорироваться.

2. I способ

Type <название типа 1> = string[10];

Кол-во символов

<название типа 2> = string;

(до 255)

Var <идентификатор>:<название типа>;

II способ

Var <идентификатор>: string[ ];

зарезерв длина

Замечание. Если две строки A и B описаны одинаково, то разрешено присвоение A:=B. Однако сравнение двух строк разрешается посимвольно. Обращение к каждому символу в строке аналогично доступу к каждому элементу в массиве.

Var A:string;

A[1] – первый символ

A[2 – второй символ

…

A[i – I – ый символ

3. Действия над строками.

a) Можно складывать, причем сложение строк неперестановочно: $a + в \neq в + a$. Сложение наз-ся конкатенация. Concat – равносильно сложению нескольких строк. Результат в 1 –ой строке concat(a1,a2,a3)

b) Присвоение

c) Разрешается сравнение строк, при этом учитывается номер каждого символа из кодовой таблицы.

Пример 'кот'<'котомка'

'котт' > 'кота'

'aa' < 'aaa'

'a' < 'я'

Все символы русского и латинского алфавитов имеют коды соответствующих их порядку, т.е. 'A' < 'B'

Разрешены процедуры и функции над строками.

1) Copy (st,p,n)

Из строки st копируется n символов в память, начиная с позиции p.

2) Delete (st,p,n)

Из строки st удаляется n символов, начиная с p.

3) Insert(st1,st2,p)

Вставить строку st1 в st2, начиная с позиции p.

4) Length (st)

Вычисляет длину строки st.

5) Pos (st1,st2)

Находит позицию первого появления строки st1 в строке st2.

6) Str(n,st)

Процедура перевода цел. Переменной n в строку st.

7) Val(st,n,m)

Процедура переводит содержимое строки st в целочисленную переменную n, причем m – номер ошибочного символа.

Пример.

Составить программу, позволяющую сформировать массив из слов данного предложения. Подсчитать количество слов, начинающихся с буквы Т.

```
uses crt;
var
  s:string;
  x:char;
  n,i,m:byte;
mas:array[1..10] of string;
begin  clrscr;
    s:='Уж небо осенью дышало';
    n:=length(s);m:=1; mas[m]:=' ';          {Обнуление нового слова}
    for i:=1 to n do
       if s[i]=' ' then begin
                          m:=m+1;
                          mas[m]:=' ';
                        end
                 else mas[m]:=mas[m]+s[i];
       n:=0;
       for i:=1 to m do
       begin
          writeln(mas[i]);
          if mas[i,1] in ['T','t'] then inc(n)
       end;
       writeln('Количество слов, начин-ся с  с T=',n);
       readln;
end.
```

Индивидуальные задания.

1. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, удалив из него все вхождения первой буквы этого слова (количество пробелов между словами не изменять).
2. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Определить количество слов, которые начинаются и заканчиваются одной и той же буквой.
3. В языке используется латинский алфавит. Глагол прошедшего времени получается из глагола настоящего времени изменением порядка следования гласных (a, o, u, i, e) на обратный. Согласные остаются на своих местах. Например, глагол padbote преобразуется в pedbota. Задается глагол настоящего времени. Преобразовать его в глагол прошедшего времени и напечатать.
4. Дана строка-предложение из символов латинского алфавита. Вывести самое короткое слово в предложении (если таких слов несколько, то вывести первое из них).

5. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Определить количество слов, которые содержат ровно три буквы 'A'.
6. Дана строка из символов латинского алфавита. Проверьте правильность расстановки тега <td>: каждому открытому тегу должен соответствовать закрытый </td>.
7. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Определить длину самого длинного слова.
8. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, но разделенные одним символом '.' (точка). В конце точку не ставить.
9. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, удалив из него все вхождения последней буквы этого слова (количество пробелов между словами не изменять).
10. Предложение состоит из слов, разделенных одним или несколькими пробелами. Написать программу, печатающую все слова, оканчивающиеся на заданный символ.
11. В предложении, состоящем из слов, разделенных одним пробелом, заменить первую букву у слов, следующих за словами die, der, das, на прописную.
12. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, удалив из него все вхождения заданного символа (количество пробелов между словами не изменять).
13. Дана строка-предложение из символов латинского алфавита. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы.
14. Дана строка-предложение из символов латинского алфавита. Вывести самое длинное слово в предложении (если таких слов несколько, то вывести последнее из них).
15. Определить, сколько раз в строке встречается заданное слово.

Лабораторная работа №10. Тип данных Record (запись).

Цель: *познакомить студентов со структурированным типом данных record, подготовиться к изучению файловых переменных.*

Для описания данных, когда с одним значением связывается набор данных разного типа, используется понятие запись. В ряде языков программирования аналогичное понятие называется структура.

Запись – это структура данных, состоящая из фиксированного числа компонентов, называемых полями записи.

В отличие от массивов компоненты (поля записи) могут быть разного типа. В качестве компонентов можно использовать данные простого и сложного типа.

2. I способ

```
Type <название типа>  = record
                        <имя компонента 1>:<тип компонента 1 >;
                        <имя компонента 2>:<тип компонента 2 >;
                        …
                        <имя компонента n>:<тип компонента n >;
                        end;
```

Один или несколько идентификаторов, которые называются полями записи

```
var
        < идентификатор >: <тип>;
```

II способ

```
var < идентификатор>: record
                        …
                        end;
```

Замечание В качестве поля (компоненты) записи может использоваться другая запись. В этом случае ее описание либо указывается сразу после двоеточия либо описана выше в разделе type.

Пример

```
type Data=record
                Day:byte;
                Month:byte;
Year:word;
                    End;
            Type Rabotnic=record
                            Dr:Data;
                            Fio:string;
                            Pol:booleon;
                            Otd:integer
                        End;
Var x,y:rabotnic;
```

3. Обращение к конкретным полям записи производится при помощи уточнения через точку.

Например

x.fio
y.otd
x.dr.day
y.dr.year

Для упрощения работы с записью в Паскале используется оператор with .. do. Он позволяет указать переменную типа запись, а затем напрямую использовать название полей записи.

Пример

```
with x, dr do …fio…
                Begin  … otd…
                        …year…
                End
```

Варианты

With x do
With x.dr do

4. В массиве GRUP(24), описывающих данные на студентов

Имя Фамилия
Дата рождения
Адрес
Телефон

Пример.

Составить программу, позволяющую вывести данные на всех студентов, затем данные на студентов 1980 года рождения и проживающих по пр.Ленина.

```
uses crt;
type STUDENT=record
        ifa:string;
        data:record
                day:byte;
```

```
                    month:byte;
                    year:word;
                    end;
                adres:string;
                tel:word
            end;
    var GRUP:array[1..24] of student;
       i:byte;
       x:STUDENT;
    begin  clrscr;
    writeln('Создание массива данных');
    for i:=1 to 3 do
       begin
          writeln('Имя,Фамилия');
          readln(x.ifa);
          writeln('День рождения');
          readln(x.data.day);
          writeln('Месяц рождения');
          readln(x.data.month);
          writeln('Год рождения');
          readln(x.data.year);
          writeln('Адрес');
          readln(x.adres);
          writeln('Телефон');
          readln(x.tel);
          GRUP[i]:=x;
          end;
    writeln('Имя Фамилия':15,'День Месяц Год Адрес':20,'Телефон');
    for i:=1 to 3 do with GRUP[i] do
    writeln(ifa:15,data.day:4,data.month:4,data.year:6,adres:15,tel:7);
    writeln;writeln('Данные студентов, рожденные после 1980 г. и
    проживающие на пр.Ленина');
    for i:=1 to 3 do
    with GRUP[i] do
    if(data.year>1980) and(copy(adres,1,9)='пр.Ленина') then
    writeln(ifa:15,data.day:4,data.month:4,data.year:6,adres:15,tel:7);
    readln;
    end.
```

Индивидуальные задания.

1. Абитуриенты

Запись: Ф.И.О, год рождения, год окончания школы, средний балл по аттестату.

Результаты обработки данных: список со средним баллом > 4.5 и количество отличников.

2. Абитуриенты

Запись: Ф.И.О, год рождения, какое учебное заведение окончил, место проживания (город, село)

Результаты обработки данных: список абитуриентов, проживающих не в Гомеле и количество окончивших техникум

3. Абитуриенты

Запись: Ф.И.О, год окончания школы, средний балл по аттестату, служба в армии

Результаты обработки данных: список окончивших школу до 1990 года и количество служивших в армии

4. Абитуриенты

Запись: Ф.И.О, год рождения, пол, средний балл по аттестату

Результаты обработки данных: список абитуриентов моложе 18 лет и количество абитуриентов мужского пола

5. Абитуриенты

Запись: Ф.И.О, год рождения, средний балл по аттестату, баллы на вступительных экзаменах (4 экзамена)

Результаты обработки данных: список абитуриентов , набравших проходной балл (18) и количество сдавших экзамены без троек

6. Абитуриенты

Запись: Ф.И.О, пол, средний балл по аттестату, баллы на вступительных экзаменах (4 экзамена)

Результаты обработки данных: список абитуриентов со средним баллом по аттестату >4.75 и количество сдавших экзамены только на "отлично"

7. Студенты

Запись: Ф.И.О, пол, место проживания (дома, в общежитии, на квартире), получает ли стипендию

Результаты обработки данных: список студентов проживающих в общежитии и количество получающих стипендию

8. Студенты

Запись: Ф.И.О, год рождения, пол, место проживания (дома, в общежитии, на квартире)

Результаты обработки данных: список студентов моложе 18 лет и процент проживающих в общежитии

9. Студенты

Запись: Ф.И.О, пол, год рождения, баллы на экзаменах (4 экзамена)

Результаты обработки данных: список студентов получивших на экзаменах 4 и 5 и процент сдавших без троек

10. Студенты

Запись: Ф.И.О, год рождения, место проживания (дома, в общежитии, на квартире), баллы на экзаменах (4 экзамена)

Результаты обработки данных: список студентов получивших на экзаменах 2 и процент сдавших с первого предъявления

11. Студенты

Запись: Ф.И.О, пол, баллы на экзаменах (4 экзамена), получает ли стипендию

Результаты обработки данных: список студентов, получающих стипендию и процент сдавших экзамены на "отлично"

12. Студенты

Запись: Ф.И.О, пол, год рождения, баллы на экзаменах (4 экзамена)

Результаты обработки данных: список студентов мужского пола и процент сдавших экзамены на 4 и 5

13. Студенты

Запись: Ф.И.О, место проживания (дома, в общежитии, на квартире), год рождения, баллы на экзаменах (4 экзамена)

Результаты обработки данных: список студентов проживающих в общежитии и процент получивших на экзамене 2

14. Студенты

Запись: Ф.И.О, место проживания (дома, в общежитии, на квартире), баллы на экзаменах (4 экзамена), получает ли стипендию

Результаты обработки данных: список студентов проживающих в общежитии и получающих стипендию процент сдавших экзамены на отлично

Учебно-методическое обеспечение к самостоятельной работе студентов по дисциплине

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

(НАИМЕНОВАНИЕ ДИСЦИПЛИНЫ)

Учебно-методические рекомендации к самостоятельной работе студентов включают:

1. комплект тестовых заданий;
2. экзаменационные вопросы;
3. контрольные вопросы к зачету с оцнкой.

Вопросы по усвоению теоретического материала и оценке практических навыков приведены в рабочей программе после каждого раздела дисциплины.

КОМПЛЕКТ ТЕСТОВОГО МАТЕРИАЛА.

1. Минимальная единица информации в компьютере:
а) Бит
б) Байт
в) Килобайт
г) Мегабайт

2. Информационная система обладает следующими свойствами:
а) Целостность и делимость
б) Целостность и неделимость
в) Ограниченность и делимость
г) Целостность и доступность

3. Выберите устройства ввода информации:
а) Клавиатура, мышь
б) Клавиатура, мышь, сканер, колонки
в) Клавиатура, мышь, сканер
г) Веб-камера, мышь, принтер

4. Антивирусные средства входят в состав:
а) Прикладного ПО
б) Базового ПО

5. MS Windows - это:
а) Однозадачная ОС
б) Многозадачная ОС
в) Сетевая ОС

6. Диалоговый, интерактивный режимы, режим реального времени - это классификация ИС:
а) По структуре аппаратных средств
б) По режиму работы
в) По характеру взаимодействия с пользователями
г) По назначению

7. К прикладному ПО относят:
а) Текстовые и графические редакторы
б) Текстовые и графические редакторы, электронные таблицы, СУБД и т.д.
в) Текстовые графические редакторы, программы контроля, интерпретаторы
г) Утилиты, электронные таблицы, СУБД

8. Коммуникационная система по сбору, передаче, переработке информации об объекте, снабжающая работника любой профессии информацией для реализации функции управления, это:
а) Данные
б) Информация
в) Информационная система
г) Информационные технологии

9. Информационно-управляющие, информационно-поисковые, информационносправочные системы, системы поддержки принятия решения - это классификация ИС:
а) По назначению
б) По структуре аппаратных средств
в) По режиму работы
г) По характеру взаимодействия с пользователями

10. Комплекс программ, предназначенный для решения задач определенного класса, это:
а) Система программирования
б) Базовое ПО
в) Пакет прикладных программ
г) Сервисное программное обеспечение

11. Совокупность систематизированных и организованных специальным образом данных и знаний:
а) Информационная среда
б) Информационная система
в) Информационные технологии
г) Информация

12. К текстовым редакторам и процессорам относят:
а) блокнот
б) электронная таблица
в) MS Word
г) СУБД
д) WordPad
е) MS Internet Explorer

13. Минимальным объектом, используемым в текстовом редакторе, является:
а) Слово
б) Пиксель
в) Абзац
г) Символ

14. При задании параметров страницы устанавливаются
а) Гарнитура, размер, начертание
б) Отступ, интервал
в) Поля, ориентация
г) Стиль, шаблон

15. В текстовом редакторе основными параметрами при задании шрифта являются:
а) Гарнитура, размер, начертание

б) Отступ, интервал
в) Поля, ориентация
г) Стиль, шаблон

16. Проверка правописания находится в меню:
а) Вставка
б) Вид
в) Рецензирование
г) Разметка страницы

17. Какого способа выравнивания нет в WORD:
а) Выравнивание по левому краю
б) Выравнивание по правому краю
в) Выравнивание по высоте
г) Выравнивание по ширине

18. С помощью этой вкладки меню создаются примечания, комментарии и библиография к курсовым и дипломным работам:
а) Ссылки
б) Вид
в) Рассылки
г) Вставка

19. К вкладке меню «Абзац» не относится:
а) Сортировка
б) Маркеры
в) Гарнитура шрифта
г) Нумерация

20. Гиперссылка находится в меню:
а) Вставка
б) Ссылки
в) Вид
г) Рецензирование

21. MS Excel - это:
а) Текстовый процессор
б) Текстовый редактор
в) Табличный процессор
г) Г рафический редактор

22. На основе чего строится любая диаграмма?
а) Книги Excel
б) Графического файла
в) Текстового файла
г) Данных таблицы

23. Какая из ссылок является абсолютной?
а) С22

б) R1C2
в) A5
г) #A#5

24. Что из перечисленного не является характеристикой ячейки?а) Имя
б) Адрес
в) Размер
г) Значение

25. Формула в электронных таблицах не может включать:
а) Имена ячеек
б) Числа
в) Текст
г) Знаки арифметических операций

26. Адрес ячейки электронной таблицы - это:
а) Любая последовательность символов
б) Номер байта оперативной памяти, отведенного под ячейку
в) Имя, состоящее из имени столбца и номера строки
г) Адрес байта оперативной памяти, отведенного под ячейку

27. Указание адреса ячейки в формуле называется:
а) Ссылкой
б) Функцией
в) Оператором
г) Именем ячейки

28. Какая из этих формул записана верно для Microsoft Excel?
а) (A5+G7)/F4
б) =(D4+44)*D3
в) =(Д4+С8)*К3
г) F(x)=A5-J6

29. Укажите, как называется программный комплекс, предназначенный для создания и обслуживания базы данных:
а) СУБД
б) MS Office
в) ИС
г) ИТ

30. Продолжите фразу: реляционная база - это та база данных, в которой информация хранится в виде:
а) Таблиц
б) Запросов
в) Отчетов
г) Списков

31. Модель БД, представляющая совокупность объектов различного уровня, причём схема связей может быть любой:
а) Сетевая

б) Иерархическая
в) Реляционная
г) Структурная

32. Основная категория объектов в реляционной СУБД:
а) Таблицы
б) Запросы
в) Формы
г) Отчёты

33. Поля таблицы СУБД MS Access не могут содержать:
а) Текст
б) Числа
в) Видеофайл
г) Даты
д) Рисунки

34. Объект БД, используемый для ввода, изменения или отображения данных из таблицы или запроса:
а) Макрос
б) Запрос
в) Форма
г) Отчет

35. Поле базы данных, это:
а) Строка таблицы
б) Столбец таблицы
в) Название таблицы
г) Свойство объекта

36. Вирус, скрывающий себя за счет шифрования основного тела вируса и существенной модификации от копии к копии модуля-расшифровщика, называется:
а) Макровирусом
б) Полиморфным
в) Троянским
г) Вирусом-спутником

37. Вирусы, которые заражают документы, созданные в приложениях Microsoft Office, используя возможности встроенных языков:
а) Макровирусы
б) Полиморфные
в) Троянские
г) Вирусы-спутники

38. Вирус, проникающий на компьютер под видом полезной программы. Не имеет собственного механизма распространения.
а) Макровирусы
б) Полиморфные
в) Троянские
г) Вирусы-спутники

39. Вирусы, которые не изменяют файл и не внедряются в него, а создают для exe-файлов файлы-спутники с расширением .com.
а) Макровирусы
б) Полиморфные
в) Троянские
г) Вирусы-спутники

40. Протоколы информационной сети, это:
а) Специализированные средства, позволяющие организовывать общение пользователей по каналам компьютерной связи
б) Совокупность правил, регулирующих порядок обмена данными в сети
в) Система передачи электронной информации, позволяющая каждому пользователю сети получить доступ к программам документам удаленного компьютера
г) Совокупность правил начинающего блоггера

41. В зависимости от конфигурации физических связей или топологии выделяют типы компьютерных сетей:
а) Ячеистая, кольцевая, шина, звезда
б) Проводные, беспроводные
в) Одноранговые, многоранговые
г) Г лобальные, региональные, локальные

42. Службой глобальной сети Интернет, предоставляющей доступ к гипертекстовой информационной системе, является...
а) E-mail
б) DNS
в) WWW
г) FTP

43. В зависимости от используемой среды передачи данных в компьютерных сетях выделяют типы сетей:
а) Проводные, беспроводные
б) Ячеистая, кольцевая, общая шина, звезда
в) Одноранговые, многоранговые
г) Глобальные, региональные, локальные

44. В зависимости от территориального расположения абонентских систем выделяют компьютерные сети:
а) Проводные, беспроводные
б) Ячеистая, кольцевая, общая шина, звезда
в) Одноранговые, многоранговые
г) Глобальные, региональные, локальные

45. Протокол TCP обеспечивает:
а) Разбиение файлов на IP-пакеты в процессе передачи и сборку файлов в процессе получения
б) Прием, передачу и выдачу одного сеанса связи
в) Доставку информации от компьютера-отправителя к компьютеру-получателю
г) Доступ пользователя к переработанной информации

46. Компьютер, подключенный к сети и предоставляющий пользователям сети определенные услуги, в т.ч. хранение данных общего пользования и печать документов
а) Рабочая станция
б) Сервер сети
в) Маршрутизатор
г) Повторитель

47. Алгоритм - это ...
а) некоторые истинные высказывания, которые должны быть направлены на достижение поставленной цели
б) отражение предметного мира с помощью знаков и сигналов, предназначенное для конкретного исполнителя
в) понятное и точное предписание исполнителю совершить определённую последовательность действий
г) представление кода программы на языке программирования
д) система инструкций для исполнителя

48. Алгоритм, записанный на «понятном» компьютеру языке программирования, называется
а) исполнителем алгоритмов
б) программой
в) блок-схема
г) текстовкой
д) протоколом алгоритма

49. Свойство алгоритма «массовость» обозначает ...
а) что команды должны следовать друг за другом
б) что каждая команда должна быть описана в расчёте на конкретного исполнителя
в) разбиение алгоритма на конечное число простых шагов
г) обязательное наличие завершающих инструкций
д) что алгоритм должен обеспечивать возможность его применения для решения однотипных задач
е) использование любым исполнителем

50. Линейный алгоритм - это ...
а) способ представления алгоритма с помощью геометрических фигур
б) набор команд, которые выполняются последовательно друг за другом
в) понятное и точное предписание исполнителю для выполнения различных ветвлений
г) последовательное выполнение команд

51. Циклический алгоритм - это алгоритм, ...
а) содержащий ветвление
б) выполняющий последовательные действия
в) представленный в графической форме
г) записанный в виде формулсодержащий многократное повторение некоторых операторов

52. Свойство алгоритма «дискретность» обозначает ...
а) что команды должны следовать друг за другом
б) что каждая команда должна быть описана в расчёте на конкретного исполнителя

в) разбиение алгоритма на конечное число простых шагов
г) обязательное наличие завершающих инструкций
д) последовательность выполнения команд алгоритма

53. Система программирования - это ...
а) машинно-зависимый язык низкого уровня, в котором короткие мнемонические имена соответствуют отдельным машинным командам
б) система для разработки новых программ на конкретном языке программирования
в) машинный язык, который понимает процессор
г) язык программирования для комбинирования компонентов, набор которых создается заранее при помощи других языков

54. Значение переменной b в результате выполнения действия операторов будет равно….?

```
a:=-1;
b:=1;
if (b>0) and (a>0) then
        b:=74
else
        if (b<0) and (a<0) then b:=87 else b:=10;
```

55. Задан двумерный массив a[1..n,1..n]. Фрагмент алгоритма

```
for i:=1 to n do
 begin
 c:=a[i,i];
 a[i,i]:=a[i,n-i+1];
a[i,n-i+1]:=c
 end;
```

а) меняет порядок строк таблицы
б) меняет порядок столбцов
в) меняет местами элементы главной и побочной диагонали
г) меняет местами элементы i-ой строки и побочной диагонали
д) ничего не меняет в таблице

56. Задан двумерный массив a[1..n,1..n]. Фрагмент алгоритма

```
for i:=1 to n do
for j:=1 to n do
begin
c:=a[i,j];
 a[i,j]:=a[n-i+1,j];
 a[n-i+1,j]:=c
end;
```

а) меняет порядок строк таблицы
б) меняет порядок столбцов
в) меняет местами элементы главной и побочной диагонали
г) меняет местами элементы i-ой строки и побочной диагонали
д) ничего не меняет в таблице

57. Дана функция:

```
 function TestDiag:boolean;
var j,s:integer;
```

```
begin s:=0;
for j:=1 to N do
s:=s+a[(N-j)*N+j];
TestDiag:=s;
end;
```

Здесь переменные j и s являются

а) глобальными
б) локальными
в) Формальными
г) фактическими параметрами.

58. Какое утверждение о цикле for верно?

а) цикл for выполняется по крайней мере один раз
б) индексная переменная всегда увеличивается на 1
в) индексная переменная не может модифицироваться внутри цикла
г) цикл for может быть неопределенным

ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ

1 СЕМЕСТР

1. Информационные технологии как система.
2. Этапы эволюции информационных технологий.
3. Двоичная система счисления.
4. Перевод целых чисел из одной системы счисления в другую.
5. Перевод дробных чисел из одной системы счисления в другую.
6. Кодирование информации в ЭВМ.
7. Сущность и назначение машинных кодов – прямой, дополнительный и обратный код.
8. Роль АБД в системах автоматизированного проектирования.
9. Этапы создания АБД.
10. Функции администратора баз данных.
11. Иерархическая модель баз данных.
12. Сетевая модель баз данных.
13. Реляционная модель баз данных.
14. Система управления базами данных.
15. Понятие алгоритма. Свойства алгоритмов.
16. Типы алгоритмов. Линейные, разветвляющиеся, алгоритмы. Способы задания разветвляющегося алгоритма на Turbo Pascal, пример.
17. Оператор множественного выбора CASE, пример.
18. Циклические алгоритмы. Виды циклов в Turbo Pascal, пример.
19. Табулирование функции одной переменной на заданном отрезке. Блок-схема, программа.
20. Вложенные циклы, пример.
21. Алгоритм: определить, является ли введенное натуральное число простым. Блок-схема, программа.
22. Нахождение конечной суммы и произведения. Блок-схема, программа.
23. Итерационные циклы. Нахождение суммы ряда с заданной точностью по общей формуле. Блок-схема, программа.
24. Итерационные циклы. Нахождение суммы ряда с заданной точностью по рекуррентной формуле. Блок-схема, программа.
25. Массивы.
26. Одномерные массивы. Ввод, вывод. Блок-схема, программа.
27. Двумерные массивы. Ввод, вывод. Блок-схема, программа.
28. Обработка массивов. Нахождение суммы, произведения элементов массива. Блок-схема, программа.
29. Нахождение максимального элемента массива. Блок-схема, программа.
30. Алгоритм слияния двух упорядоченных массивов. Блок-схема, программа.
31. Алгоритм умножения матрицы на вектор. Блок-схема, программа.
32. Алгоритм умножения матрицы на матрицу. Блок-схема, программа.
33. Сортировка, основные понятия. Эффективность сортировок.
34. Сортировка массивов. Метод прямого выбора. Блок-схема, программа.
35. Сортировка массивов. Метод прямого обмена. Блок-схема, программа.
36. Сортировка массивов. Улучшения метода прямого обмена. Шейкерная сортировка. Блок-схема, программа.
37. Сортировка массивов. Метод прямого включения. Блок-схема, программа.
38. Линейный поиск. Блок-схема, программа.
39. Двоичный поиск. Блок-схема, программа.

40. Основные этапы развития программирования как науки. Стихийное программирование.
41. Структурный подход к программированию.
42. Объектный подход к программированию.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ЗАЧЕТУ С ОЦЕНКОЙ

1. Использование процедур и функциий.
2. Формальные и фактические параметры, локальные и глобальные переменные.
3. Процедуры обработки символьной информации.
4. Процедуры работы со строками.
5. Файловая организация ввода-вывода информации.
6. Записи, приемы работы с записями.
7. Алгоритмы одномерной оптимизации.
8. Принципы построения вычислительных сетей.
9. Способы коммутации и передачи данных.
10. Основные компоненты и типы ЛВС.
11. Компоновка локальных вычислительных сетей.
12. Физическая среда передачи данных.
13. Кодирование сигналов и асинхронная передача.
14. Беспроводные сети.
15. Организация обмена информацией в ЛВС.
16. Характеристика основных информационных ресурсов Интернет.
17. Принципы функционирования Интернет.
18. Характеристика и типы модемов.
19. Направления исследований в области искусственного интеллекта.
20. Машинный интеллект и робототехника.
21. Характеристика поколений роботов.
22. Характеристика нейронных сетей.
23. Эвристическое программирование и моделирование.
24. Характеристика экспертных систем, основные виды деятельности ЭС.
25. Структура экспертных систем.
26. Режимы использования ЭС и инструментальные средства ЭС.
27. Основные виды деятельности ЭС.
28. Мультимедиа-технологии.
29. Геоинформационные технологии.
30. CASE –технологии.
31. Организационные и организационно-технические меры защиты информации.
32. Технические средства защиты информации.
33. Приемы и способы перехвата информации.
34. Программно-аппаратные комплексы в теплоэнергетике.