Семинарские занятия

История и перспективы развития Информатики и вычислительной техники

Простейшие ручные приспособления

История компьютера тесным образом связана с попытками человека облегчить, автоматизировать большие объёмы вычислений. Даже простые арифметические операции с большими числами затруднительны для человеческого мозга. Поэтому уже в древности появилось устройство – **абак.** **Абак** (греч. αβαξ, abákion, лат. abacus − доска) − это счётная доска, простейшее счётное устройство, применявшееся для арифметических вычислений приблизительно с IV века до н.э. в Древней Греции, Древнем Риме. В Европе абак применялся до XVIII века.

Простейший абак - это доска с прорезанными в ней желобами. Чтобы найти **сумму** двух чисел (например, 258 и 125), счетчик сначала обозначал на абаке первое слагаемое. Для этого он укладывал в нижнем желобе 8 камешков, в следующем желобе - 5 камешков и 2 камешка - в третьем желобе. Если в каком-то разряде в числе стоял нуль, то соответствующий желоб оставался пустым. Дальше счетчик добавлял в последний желоб к имеющимся там 8 камешкам еще 5, затем вынимал оттуда 10 (там оставалось 3) и 1 камешек добавлял во второй желоб. Потом добавлял во второй желоб еще 2 камешка и 1 камешек - в третий желоб, после этого камешки на доске показывали число 383.

Рисунок 1 Реконструкция римского абака

Абаки использовались уже в V-IV веках до нашей эры. Их изготавливали из бронзы, камня, слоновой кости, цветного стекла. Слово «абак» имеет греческое происхождение и буквально означает «пыль», хотя его смысловое значение - «счетная доска». В чем тут дело? Ответ прост: изначально камешки раскладывали на совершенно ровной доске, а, чтобы они не скатывались со своего первоначального положения, доска покрывалась тонким слоем песка или пыли. А от слова «камешек» (по латыни - «calculus») произошло название современного счетного прибора - **калькулятор**.

В ***России*** ещё в средние века (16-17 вв.) на основе абака было разработано другое приспособление – ***русские счёты***. Десятичный абак, или русские счёты, в которых используется десятичная система счисления и возможность оперировать четвертями, десятыми и сотыми дробными долями появились в России на рубеже XV - XVI веков и активно применялись в торговле вплоть до последнего десятилетия XX века. От классического абака счёты отличаются увеличением разрядности каждого числового ряда и конструкцией. Ещё одна

характерная особенность русских счёт - специально выделенный разряд для счёта в четвертях. С момента своего возникновения счёты практически не изменились.

Рисунок 2 Счеты

С появлением дешёвых электронных калькуляторов счёты практически полностью вышли из употребления. Ещё раньше, в начале 1980-х годов, обучение пользованию счётами было исключено в СССР из школьной программы.

Механические приспособления

Механизация вычислительных операций началась в XVII веке. На первом этапе для создания механических вычислительных устройств использовались механизмы, аналогичные часовым.

В 1623 год − немецкий ученый Вильгельм Шиккард разработал первое в мире механическое устройство («суммирующие часы») для выполнения операций сложения и вычитания шестиразрядных десятичных чисел. Было ли устройство реализовано при жизни изобретателя, достоверно неизвестно, но в 1960 году оно было воссоздано по чертежам и подтвердило свою работоспособность.

В 1642 году французский механик Блез Паскаль сконструировал первое в мире механическое цифровое вычислительное устройство («Паскалин»), построенное на основе зубчатых колес. Оно могло суммировать и вычитать пятиразрядные десятичные числа, а последние модели оперировали числами с восемью десятичными разрядами.

В 1673 г. немецкий философ и математик Готфрид Вильгельм Лейбниц создал механический калькулятор, который при помощи двоичной системы счисления выполнял умножение, деление, сложение и вычитание. Операции умножения и деления выполнялись путём многократного повторения операций сложения и вычитания.

Однако широкое распространение вычислительные аппараты получили только в 1820 году, когда француз Чарльз Калмар изобрёл машину, которая могла производить четыре основных арифметических действия. Машину Калмара назвали арифмометр. Благодаря своей универсальности арифмометры

использовались довольно длительное время до 60-х годов XX века.

Арифмометр Однера - успешная разновидность арифмометров, разработанная российским механиком шведского происхождения В. Т. Однером.

Рисунок 3 Арифмометр Однера

Однер заинтересовался арифмометрами в 1871 году после ремонта случайно попавшего к нему арифмометра Тома де Кольмара - единственного серийного арифмометра тех лет. Уже в 1873 году был построен первый прототип, а в 1877 изготовлены 14 экземпляров по заказу Людвига Нобеля. В 1878-1890 годах Однер совершенствовал и запатентовал свою машину в нескольких странах.

В 1890 году было открыто производство в России, в 1891 году - производство в Германии.

После октября 1917 года, когда предприятие Однера было национализировано, наследники Однера репатриировались в Швецию и создали новое производство, продавая арифмометры под торговой маркой Original-Odhner («подлинный Однер»).

В 1924 году старый Петербургский завод Однера был перенесен в Москву

и продолжил выпуск клона арифмометра Однера под торговой маркой «Феликс».

«Феликс» - самый распространённый в СССР арифмометр. Выпускался с 1929 по 1978 годы общим тиражом несколько миллионов машин. Всего было создано более двух десятков модификаций арифмометра. Основными производителями являлись заводы счётных машин в Курске («Счётмаш»), в Пензе (Пензенский завод вычислительной техники) и в Москве (Завод счётно-аналитических машин имени В. Д. Калмыкова.

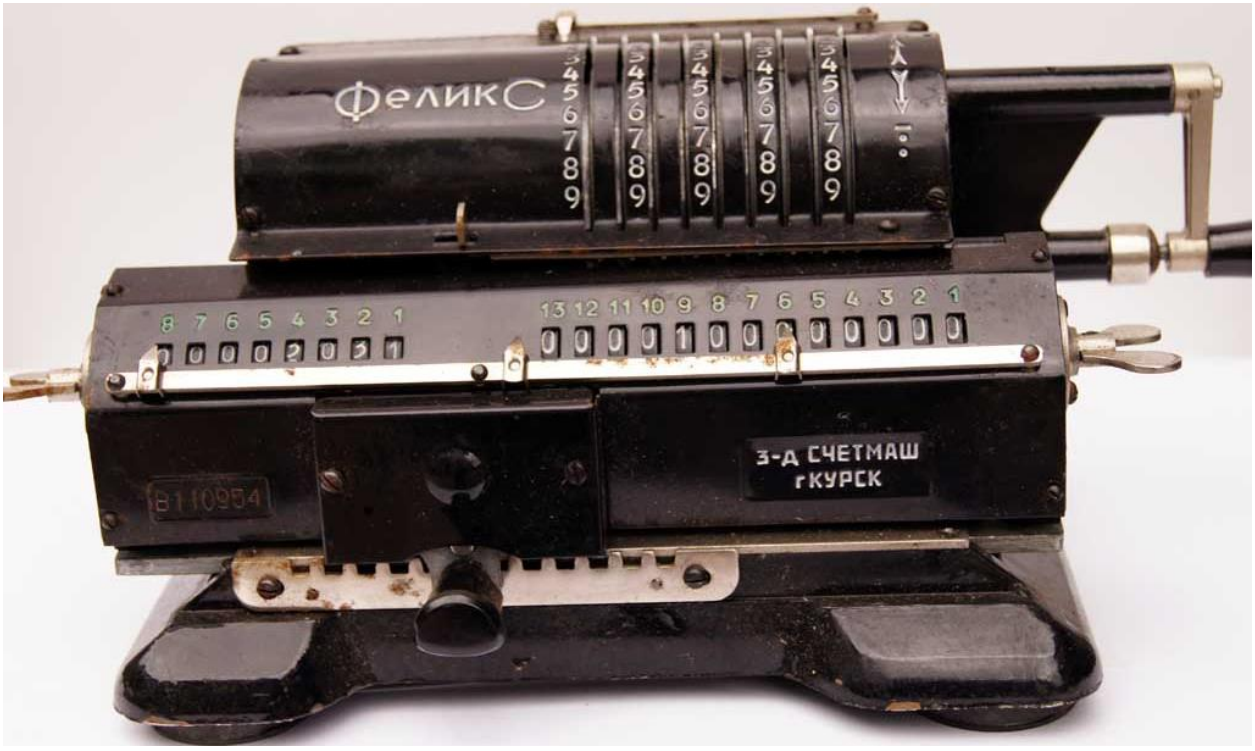

Рисунок 4 Арифмометр Феликс

Эта счётная машина является модификацией рычажного арифмометра Однера. Она позволяет работать с операндами длиной до 9 знаков и получать ответ длиной до 13 знаков (до 8 для частного от деления). Ввод чисел осуществляется перемещением рычажков вверх-вниз. Операция сложения требует оттягивания расположенной справа ручки и проворачивания её на один оборот на себя. Операция вычитания - наоборот, проворачивания на один оборот от себя.

В арифмометре использован очень простой и в то же время надёжный транспортный механизм каретки, отличавший его от всех западных аналогов.

Вес устройства составляет 4-6 кг.

Автоматизация вычислений

Идея автоматизации вычислительных операций пришла из часовой промышленности. Старинные монастырские башенные часы были построены так, чтобы в заданное время включать механизм, связанный с системой колоколов.

В **1833 году** английский ученый, профессор Кембриджского университета **Чарльз Беббидж** разработал проект **аналитической машины**, которая имела черты современного компьютера. Это был гигантский арифмометр с программным управлением, арифметическим и запоминающим устройствами. Оно имело устройство для ввода информации, блок управления, запоминающее устройство и устройство вывода результатов.

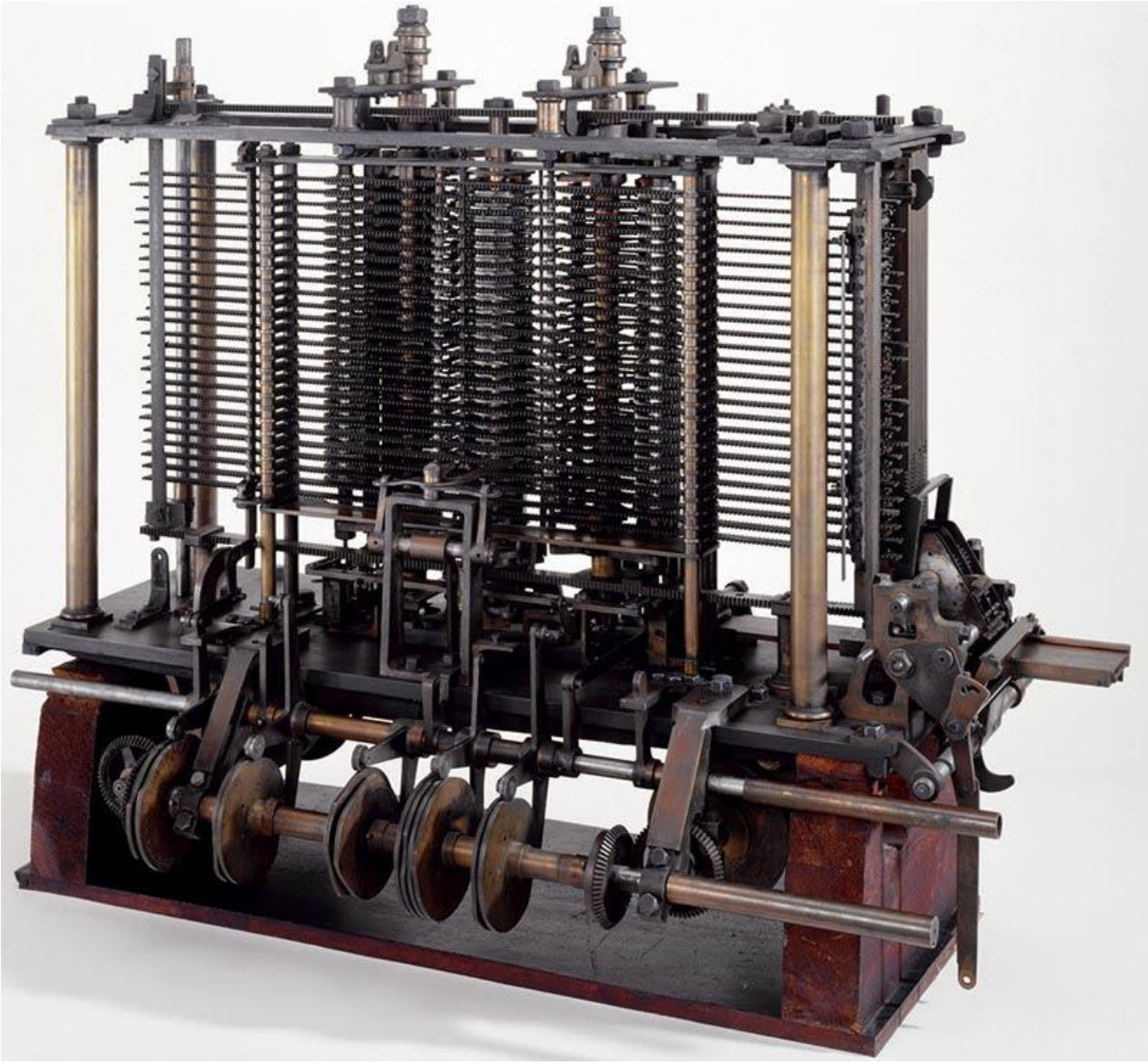

Рисунок 5 Модель «Аналитической машины» Бэббиджа, созданная по проекту автора

Сотрудницей и помощницей Ч. Беббиджа во многих его научных изысканиях была леди **Ада Лавлейс** (урожденная Байрон).

Она разработала первые программы для машины и предвосхитила основы современного программирования для цифровых вычислительных машин с программным управлением. Заложила многие идеи и ввела ряд понятий и терминов, сохранившихся до настоящего времени.

Она предсказала появление современных компьютеров как многофункциональных машин не только для вычислений, но и для работы с

графикой, звуком. В середине 70-х годов двадцатого столетия министерство обороны США официально утвердило название единого языка программирования американских вооруженных сил. Язык носит название **Ada**. День программиста отмечается в день рождения Ады Лавлейс 10 декабря.

Особенностью **Аналитической машины** стало то, что здесь впервые был реализован принцип разделения информации на ***команды и данные***. Для ввода и вывода данных Бэббидж предлагал использовать перфокарты-листы из плотной бумаги с информацией, наносимой с помощью отверстий.

В **1888** году американский инженер **Герман Холлерит** сконструировал первую электромеханическую счётную машину. Эта машина, названная **табулятором**, могла считывать и сортировать статистические записи, закодированные на перфокартах. Для работы этой машины использовалось электричество. В **1890** изобретение Холлерита было использовано в 11-ой американской переписи населения. Работа, которую 500 сотрудников выполняли в течение семи лет, Холлерит с 43 помощниками на 43 табуляторах выполнил за один месяц.

Дальнейшее развитие науки и техники позволили в **1940-х годах** построить первые вычислительные машины. В **1944 г.** американский инженер **Говард Эйкен** при поддержке фирмы Ай-Би-Эм (IBM) сконструировал компьютер для выполнения баллистических расчетов. Этот компьютер, названный «**Марк 1**», по площади занимал примерно половину футбольного поля и включал более 800 километров проводов, около 750 тыс.деталей, 3304 реле. «**Марк-1**» был основан на использовании **электромеханических реле** и оперировал десятичными числами, закодированными на **перфоленте**. Машина могла манипулировать числами длиной до 23 разрядов. Для перемножения двух 23-разрядных чисел ей было необходимо 4 секунды.

Но электромеханические реле работали недостаточно быстро. В **1946 г.** По заказу Армии США был создан первый широкомасштабный электронный

цифровой компьютер **ЭНИАК** (ENIAC - электронный числовой интегратор и вычислитель), который можно было перепрограммировать для решения полного диапазона задач. Разработали его американские ученые **Джон Уильям Мокли** и **Джон Преспер Экерт.** В ЭНИАКе в качестве основы компонентной базы электромеханические реле были заменены **вакуумными лампами**. Всего комплекс включал 17468 ламп, 7200 кремниевых диодов, 1500 реле, 70000 резисторов и 10000 конденсаторов. Потребляемая мощность – 150 кВт по тем временам было достаточно для освещения большого города. Вычислительная мощность – 300 операций умножения или 5000 операций сложения в секунду. Вес – 27 тонн, более 30 метров. Вычисления проводились в десятичной системе. ЭНИАК использовался для расчета баллистических таблиц, предсказания погоды, расчетов в области атомной энергетики, аэродинамики, изучения космоса.

В СССР вычислительная машина **МЭСМ** (малая электронная счётная машина) была создана в **1951 году** под руководством академика **Сергея Алексеевича Лебедева.** Машина вычисляла факториалы натуральных чисел и решала уравнения параболы. Одновременно Лебедев работал над созданием **БЭСМ** - быстродействующей электронной счётной машины, разработка которой была завершена в 1953 году.

В **1971 году** фирмой Intel (США) был создан первый микропроцессор - программируемое логическое устройство, изготовленное по технологии СБИС (сверхбольших интегральных схем).

В **1964г.** сотрудник Стэнфордского исследовательского центра **Дуглас Энгельбарт** продемонстрировал работу первой **мыши-манипулятора**, но только четыре года спустя мышка была показана на компьютерной конференции в Сан-Франциско.

Первый персональный компьютер (ПК) в **1976г.** выпустила фирма **Apple**; в СССР ПК появились в **1985г**.

Поколения ЭВМ

ЭВМ принято делить на поколения. Для компьютерной техники характерна прежде всего быстрота смены поколений - за её короткую историю развития уже успели смениться четыре поколения и сейчас мы работаем на компьютерах пятого поколения. Определяющими признаками при отнесении ЭВМ к тому или иному поколению являются их элементная база (из каких в основном элементов они построены), быстродействие, емкость памяти, способы управления и переработки информации.

Первое поколение. 1950-1960-е годы

Компьютеры на **электронных вакуумных лампах** (диодах и триодах), а в качестве оперативных запоминающих устройств использовались электронно-лучевые трубки, в качестве внешних запоминающих устройств применялись накопители на магнитных лентах, перфокартах, перфолентах и штекерные коммутаторы.

Программирование работы ЭВМ этого поколения выполнялось в двоичной системе счисления на машинном языке, то есть программы были жестко ориентированы на конкретную модель машины.

Машины предназначались для решения сравнительно несложных научно-технических задач. Они были значительных размеров, потребляли большую мощность, имели невысокую надежность работы.

Быстродействие их не превышало 2-3 тысяч операций в секунду, емкость оперативной памяти - 2048 машинных слов длиной 48 двоичных знаков. Использовались в основном для научных расчетов.

В конце этого периода стали выпускаться устройства памяти на магнитных сердечниках.

ЭНИАК, МЭСМ, БЭСМ и первые модели ЭВМ "Минск" и "Урал".

Второе поколение ЭВМ. 1960-1970-е годы

Элементной базой машин этого поколения были полупроводниковые элементы (транзисторы). Транзисторы (твердые диоды и триоды) заменили электронные лампы в процессорах, а ферритовые (намагничиваемые) сердечники – электронно-лучевые трубки в оперативных запоминающих устройствах. Машины предназначались для решения различных трудоемких научно-технических задач, а также для управления технологическими процессами в производстве.

Появление полупроводниковых элементов в электронных схемах существенно увеличило емкость оперативной памяти, надежность и быстродействие ЭВМ. Уменьшились размеры, масса и потребляемая мощность.

Скорость ЭВМ возросла до сотен тысяч операций в секунду, а память – до десятков тысяч машинных слов. Создаются долговременные запоминающие устройства на **магнитных лентах**. Начали применять языки программирования высокого уровня, такие как **Фортран**.

В 1964 году появился первый монитор для компьютеров - IBM 2250. Это был монохромный дисплей с экраном 12 × 12 дюймов и разрешением 1024 × 1024 пикселов. Он имел частоту кадровой развертки 40 Гц.

Третье поколение ЭВМ: 1970-1980-е годы

Элементная база ЭВМ - **малые интегральные схемы** (МИС), что привело к дальнейшему увеличению скорости до миллиона операций в секунду и памяти до сотен тысяч слов. Машины предназначались для широкого использования в различных областях науки и техники.

ЭВМ третьего поколения также характеризуется крупнейшими сдвигами в архитектуре ЭВМ, их программном обеспечении, организации взаимодействия

человека с машиной. Это, прежде всего наличие развитой конфигурации внешних устройств (алфавитно-цифровые терминалы, графопостроители, магнитные диски (30 см в диаметре) и т.п.), развитая операционная система.

В период машин третьего поколения произошел крупный сдвиг в области применения ЭВМ. Если раньше ЭВМ использовались в основном для научно-технических расчетов, то в 60-70-е годы первое место стала занимать обработка символьной информации, в основном экономической.

IV поколение. 1980-1990-е годы

Переход к машинам четвертого поколения – ЭВМ на ***больших интегральных схемах (БИС)*** – происходил во второй половине ***70-х годов*** и завершился приблизительно к ***1980 г.*** Теперь на одном кристалле размером 1 $см^2$ стали размещаться сотни тысяч электронных элементов. Скорость и объем памяти возросли в десятки тысяч раз по сравнению с машинами первого поколения и составили примерно 10^9 операций в секунду и 10^7 слов соответственно.

Наиболее крупным достижением, связанным с применением БИС, стало создание ***микропроцессоров***, а затем на их основе ***микро-ЭВМ***. Если прежние поколения ЭВМ требовали для своего расположения специальных помещений, системы вентиляции, специального оборудования для электропитания, то требования, предъявляемые к эксплуатации микро-ЭВМ, ничем не отличаются от условий эксплуатации бытовых приборов. При этом они имеют достаточно высокую производительность, экономичны в эксплуатации и дешевы.

Микро-ЭВМ используются в измерительных комплексах, системах числового программного управления, в управляющих системах различного назначения.

Дальнейшее развитие микро-ЭВМ привело к созданию ***персональных компьютеров (ПК),*** широкое распространение которых началось с ***1975 г.,*** когда

фирма ***IBM*** выпустила свой первый персональный компьютер ***IBM PC.***

В период машин четвертого поколения стали также серийно производиться супер-ЭВМ. В нескольких серийных моделях была достигнута производительность свыше 1 млрд. операций в секунду.

К числу наиболее значительных разработок четвертого поколения относится ЭВМ «***Крей-3***».

Примером отечественной суперЭВМ является многопроцессорный вычислительный комплекс «***Эльбрус***».

V поколение. 1990-настоящее время

С 90-х годов в истории развития вычислительной техники наступила пора пятого поколения. Высокая скорость выполнения арифметических вычислений дополняется высокими скоростями логического вывода.

Сверхбольшие интегральные схемы повышенной степени интеграции, использование оптоэлектронных принципов (лазеры, голография).

Способны воспринимать информацию с рукописного или печатного текста, с бланков, с человеческого голоса, узнавать пользователя по голосу, осуществлять перевод с одного языка на другой. Используются модели и средства, разработанные в области искусственного интеллекта. Архитектура содержит несколько блоков: ***блок общения*** – обеспечивает интерфейс между пользователем и ЭВМ на естественном языке; ***база знаний*** – хранятся знания, накопленные человечеством в различных предметных областях; ***решатель*** - организует подготовку программы решения задачи на основании знаний, получаемых из базы знаний и исходных данных, полученных из блока общения. ***Ядро*** вычислительной системы составляет ЭВМ высокой производительности.

В связи с появлением новой базовой структуры ЭВМ в машинах пятого поколения широко используются модели и средства, разработанные в области ***искусственного интеллекта.***

Показатель	Поколения ЭВМ				
	Первое **1950-1960-е годы**	Второе **1960-1970-е годы**	Третье **1970-1980-е годы**	Четвертое **1980-1990-е годы**	Пятое **1990-настоящее время**
Элементная база процессора	Электронные лампы	Полупроводники (Транзисторы)	Малые интегральные схемы (МИС)	Большие ИС (БИС) и Сверхбольшие ИС (СБИС)	Оптоэлектроника Криоэлектроника (лазеры, голография)
Элементная база ОЗУ	Электронно-лучевые трубки	Ферритовые сердечники	Кремниевые кристаллы	БИС и СБИС	СБИС
Основные устройства ввода	Пульт, перфокарточный, перфоленточный ввод	Алфавитно-цифровой дисплей, клавиатура		Цветной графический дисплей, клавиатура, “мышь” и др.	Цветной графический дисплей, сканер, клавиатура, устройства голосовой связи с ЭВМ
Основные устройства вывода	Алфавитно-цифровое печатающее устройство (АЦПУ), перфоленточный вывод		Графопостроитель, принтер		
Внешняя память	Магнитные ленты, барабаны, перфоленты, перфокарты	Магнитный диск	Перфоленты, магнитный диск (30 см в диаметре)	Магнитные и оптические диски	
Максимальная емкость ОЗУ, байт	10^1	10^2	10^4	10^5 - 10^7	10^8 (?)
Максимальное быстродействие процессора (оп/с)	10^4	10^6	10^7	10^8 - 10^9 +Многопроцессорность	10^{12} +Многопроцессорность
Языки программирования	Универсальные языки программирования, трансляторы (машинный код)	Пакетные операционные системы, оптимизирующие трансляторы (Ассемблер, Фортран)	Процедурные языки высокого уровня (ЯВУ)	Новые процедурные ЯВУ и Непроцедурные ЯВУ	Новые непроцедурные ЯВУ
Цель использования ЭВМ	Научно-технические расчеты	Технические и экономические расчеты	Управление и экономические расчеты	Телекоммуникации, информационное обслуживание	Использование элементов искусственного интеллекта и распознавание зрительных и звуковых образов

Классификация ЭВМ

Существует достаточно много систем классификации по различным признакам.

I. Классификация по назначению:

1) СуперЭВМ предназначены для решения крупномасштабных вычислительных задач, для обслуживания крупнейших информационных банков данных. Это очень мощные компьютеры с производительностью свыше 100 мегафлопов (1 мегафлоп - миллион операций с плавающей точкой в секунду). Они называются сверхбыстродействующими. Эти машины представляют собой многопроцессорные и (или) многомашинные комплексы, работающие на общую память и общее поле внешних устройств. Различают суперкомпьютеры среднего класса, класса выше среднего и переднего края (high end).

2) Большие ЭВМ - для комплектования ведомственных, территориальных и региональных вычислительных центров. **Мэйнфреймы** предназначены для решения широкого класса научно-технических задач и являются сложными и дорогими машинами. Их целесообразно применять в больших системах при наличии не менее 200 - 300 рабочих мест.

3) Средние ЭВМ - широкого назначения для управления сложными технологическими производственными процессами. ЭВМ этого типа могут использоваться и для управления распределенной обработкой информации в качестве сетевых серверов.

4) Персональные и профессиональные ЭВМ, позволяющие удовлетворять индивидуальные потребности пользователей. На базе этого класса ЭВМ строятся автоматизированные рабочие места (АРМ) для специалистов различного уровня.

5) Встраиваемые микропроцессоры, осуществляющие автоматизацию управления отдельными устройствами и механизмами.

II. Классификация ПК по типоразмерам:

1) **Настольные** (desktop) - используются для оборудования рабочих мест, отличаются простотой изменения конфигурации. Наиболее распространены.
2) **Портативные** – удобны для транспортировки, можно работать при отсутствии рабочего места.

Основные разновидности портативных компьютеров:

Laptop (наколенник, от **lap>** - колено и **top** - поверх). По размерам близок к обычному портфелю. По основным характеристикам (быстродействие, память) примерно соответствует настольным ПК. Сейчас компьютеры этого типа уступают место ещё меньшим.

Notebook (блокнот, записная книжка). По размерам он ближе к книге крупного формата. Имеет вес около 3 кг. Является переносным персональным компьютером. Он имеет компактные габариты и встроенные аккумуляторы, позволяющие работать без сетевого напряжения.

Palmtop (наладонник) - это самый маленький ПК. Он не имеет внешней памяти на магнитных дисках, она заменена на энергозависимую электронную память. Эта память может перезаписываться при помощи линии связи с настольным компьютером. Карманный компьютер можно использовать как словарь-переводчик или записную книжку

III. Классификация по условиям эксплуатации:

По условиям эксплуатации компьютеры делятся на два типа:

1) **офисные** (универсальные) – на их основе можно собирать вычислительные системы произвольного состава;
2) **специализированные** – предназначены для решения конкретного круга задач (например, бортовые компьютеры автомобилей, самолетов).

Основные принципы функционирования ПК

Исторически компьютер появился как машина для вычислений и назывался электронной вычислительной машиной – ЭВМ. Общие принципы работы универсальных вычислительных устройств были сформулированы известным

американским математиком **Джоном фон Нейманом** в **1946** году:

1. Любая ЭВМ для выполнения своих функций должна иметь минимальный ***набор функциональных блоков***:
    - **АЛУ** – арифметическое логическое устройство. Преобразует информацию, выполняя сложение, вычитание и основные логические операции «И», «ИЛИ», «НЕ».
    - **УУ** – устройство управления. Организует процесс выполнения программ.
    - **ОЗУ** – оперативное запоминающее устройство (память), состоящее из перенумерованных ячеек. Хранит данные, адреса и команды, обладает высокой скоростью записи и чтения чисел.
    - **УВВ** – устройство ввода-вывода. Получают информацию извне, выводят её получателю.

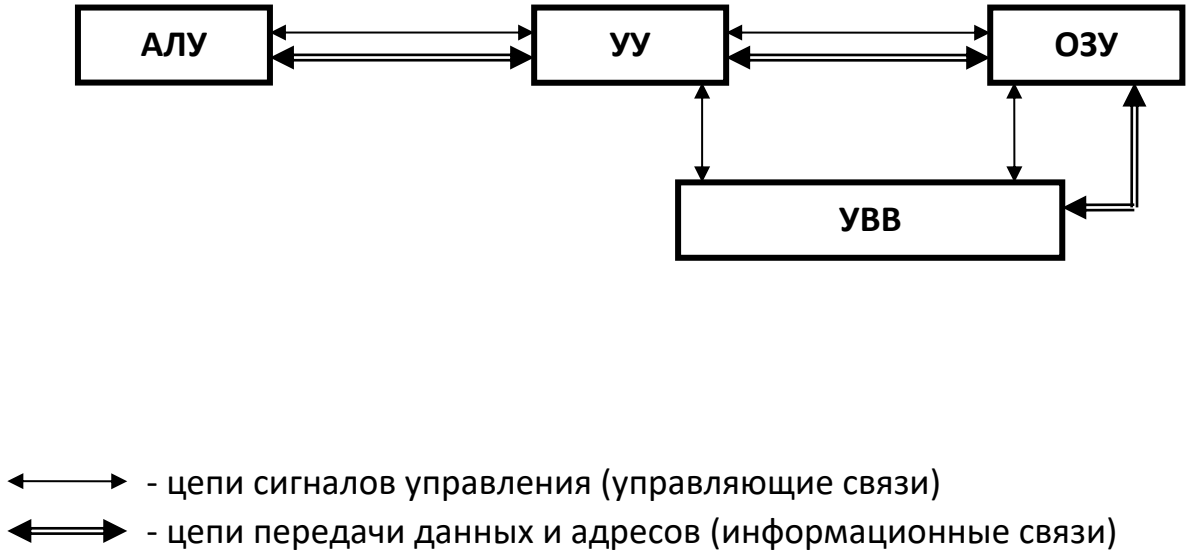

Рисунок 6 Структурная схема вычислительного устройства

Это классическая структура вычислительной машины, на основе которой уже более полувека создаются ЭВМ.

В современных компьютерах объединены **АЛУ** и **УУ** в одной сверхбольшой интегральной схеме (микропроцессор). Уменьшение габаритов **ОЗУ** позволило разместить **микропроцессор** и **ОЗУ** на одной электронной

плате (материнская). Все связи между отдельными устройствами объединены в пучок параллельных проводов (системная шина).

2. Информация кодируется в двоичной форме.
3. Алгоритм представляется в форме последовательности команд, совокупность которых называется **программой**.
4. Программы и данные хранятся в одной и той же памяти.

История программирования

Одной из самых революционных идей, приведших к созданию автоматических цифровых вычислительных машин, была высказанная в 20-х годах 19 века Чарльзом Бебиджем мысль о предварительной записи порядка действия машины для последующей автоматической реализации вычислений – программе. И, хотя использованная Бебиджем запись программы на перфокартах, придуманная для управления такими станками французским изобретателем Жозефом Мари Жаккаром, технически не имеет ничего общего с современными приемами хранения программ в ПК, принцип здесь по существу один.

С этого момента начинается история программирования. Аду Левлейс, современницу Бебиджа, называют первым в мире программистом. Она теоретически разработала некоторые приемы управления последовательностью вычислений, которые используются в программировании и сейчас. Ею же была описана и одна из важнейших конструкций практически любого современного языка программирования – цикл.

Революционным моментом в истории языков программирования стало появление системы кодирования машинных команд с помощью специальных символов, предложенной Джоном Моучли. Система кодирования, предложенная им, вдохновила одну из его сотрудниц Грейс Мюррей Хоппер. При работе на компьютере «Марк-1» ей и ее группе пришлось столкнуться со

многими проблемами и все, что ими придумано, было впервые. В частности, они придумали подпрограммы. И еще одно фундаментальное понятие техники программирования впервые ввели Хоппер и ее группа – «отладка». В конце 40-х годов Дж. Моучли создал систему под названием «Short Code», которая являлась примитивным языком программирования высокого уровня. В ней программист записывал решаемую задачу в виде математических формул, а затем, используя специальную таблицу, переводил символ за символом, преобразовывал эти формулы в двухлитерные коды. В дальнейшем специальная программа компьютера превращала эти коды в двоичный машинный код. Система, разработанная Дж. Моучли, считается одним из первых примитивных интерпретаторов.

Уже в 1951 г. Хоппер создала первый в мире компилятор и ею же был введен сам этот термин. Компилятор Хоппер осуществлял функцию объединения команд и в ходе трансляции производил организацию подпрограмм, выделение памяти компьютера, преобразование команд высокого уровня (в то время псевдокодов) в машинные команды. «Подпрограммы находятся в библиотеке (компьютера), а когда вы подбираете материал из библиотеки – это называется компиляцией» – так она объясняла происхождение введенного ею термина. В 1954 году группа под руководством Г. Хоппер разработала систему, включающую язык программирования и компилятор, которая в дальнейшем получила название Math-Matic. После удачного завершения работ по созданию Math-Matic Хоппер и ее группа принялись за разработку нового языка и компилятора, который позволил бы пользователями программировать на языке, близком к обычному английскому. В 1958 г. появился компилятор Flow-Matic. Компилятор Flow-Matic был первым языком для задач обработки коммерческих данных.

Разработки в этом направлении привели к созданию языка Кобол (COBOL – Common Business Oriented Language). Он был создан в 1960 году. В этом языке по сравнению с Фортраном и Алголом, слабее развиты математические

средства, но зато хорошо развиты средства обработки текстов, организация вывода данных в форме требуемого документа. Он задумывался как основной язык для массовой обработки данных в сферах управления и бизнеса.

Середина 50-х годов характеризуется стремительным прогрессом в области программирования. Роль программирования в машинных командах стала уменьшаться. Стали появляться языки программирования нового типа, выступающие в роли посредника между машинами и программистами. Первым и одним из наиболее распространенных был Фортран (FORTRAN, от FORmula TRANslator – переводчик формул), разработанный группой программистов фирмы IBM в 1954 году (первая версия). Этот язык был ориентирован на научно-технические расчеты математического характера и является классическим языком программирования при решении на ПК математических и инженерных задач.
Для первых языков программирования высокого уровня предметная ориентация языков была характерной чертой.

Особое место среди языков программирования занимает Алгол, первая версия которого появилась в 1958 году. Одним из разработчиков Алгола был «отец» Фортрана Джон Бэкус. Название языка ALGOrithmic Language подчеркивает то обстоятельство, что он предназначен для записи алгоритмов. Благодаря четкой логической структуре Алгол стал стандартным средством записи алгоритмов в научной и технической литературе.

В середине 60-х годов Томас Курц и Джон Камени (сотрудники математического факультета Дартмунтского колледжа) создали специализированный язык программирования, который состоял из простых слов английского языка. Новый язык назвали «универсальным символическим кодом для начинающих» (Beginner All-Purpose Symbolic Instruction Code, или, сокращенно, BASIC). Годом рождения нового языка можно считать 1964. Сегодня универсальный язык Бейсик (имеющий множество версий) приобрел большую популярность и получил широкое распространение среди

пользователей ПК различных категорий во всем мире. В значительно мере этому способствовало то, что Бейсик начали использовать как встроенный язык персональных компьютеров, широкое распространение которых началось в конце 70-х годов. Однако Бейсик неструктурный язык, и поэтому он плохо подходит для обучения качественному программированию. Справедливости ради следует заметить, что последние версии Бейсика для ПК (например, QBasic) стали более структурными и по своим изобразительным возможностям приближаются к таким языкам, как Паскаль.

Разработчики ориентировали языки на разные классы задач, в той или иной мере привязывали их к конкретной архитектуре ПК, реализовывали личные вкусы и идеи. В 60-е годы были предприняты попытки преодолеть эту «разноголосицу» путем создания универсального языка программирования. Первым детищем этого направления стал PL/1 (Programm Language One), разработанный фирмой IBM в 1967 году. Этот язык претендовал на возможность решать любые задачи: вычислительные, обработки текстов, накопления и поиска информации. Однако он оказался слишком сложным, транслятор с него – недостаточно оптимальным и содержал ряд невыявленных ошибок.

Однако линия на универсализацию языков была поддержана. Старые языки были модернизированы в универсальные варианты: Алгол-68 (1968 г.), Фортран-77. Предполагалось, что подобные языки будут развиваться и усовершенствоваться, станут вытеснять все остальные. Однако ни одна из этих попыток не увенчалась успехом.

Язык ЛИСП появился в 1965 году. Основным в нем служит понятие рекурсивно определенных функций. Поскольку доказано, что любой алгоритм может быть описан с помощью некоторого набора рекурсивных функций, то ЛИСП по сути является универсальным языком. С его помощью ПК может моделировать достаточно сложные процессы, в частности – интеллектуальную деятельность людей.

Пролог разработан во Франции в 1972 году для решения проблем «искусственного интеллекта». Пролог позволяет в формальном виде описывать различные утверждения, логику рассуждений и заставляет ПК давать ответы на заданные вопросы.

Значительным событием в истории языков программирования стало создание в 1971 году языка Паскаль. Его автор – швейцарский ученый Никлаус Вирт. Вирт назвал его в честь великого французского математика и религиозного философа XVII века Блеза Паскаля, который изобрел первое суммирующее устройство, именно поэтому новому языку было присвоено его имя. Этот язык первоначально разрабатывался как учебный язык структурного программирования, и, действительно, сейчас он является одним из основных языков обучения программированию в школах и вузах.

В 1975 году два события стали вехами в истории программирования – Билл Гейтс и Пол Аллен заявили о себе, разработав свою версию Бейсика, а Вирт и Йенсен выпустили классическое описание языка «Pascal User Manual and Report».

Не менее впечатляющей, в том числе и финансовой, удачи добился Филип Кан, француз, разработавший в 1983 году систему Турбо-Паскаль. Суть его идеи состояла в объединении последовательных этапов обработки программы – компиляции, редактирования связей, отладки и диагностики ошибок – в едином интерфейсе. Турбо-Паскаль – это не только язык и транслятор с него, но еще и операционная оболочка, позволяющая пользователю удобно работать на Паскале. Этот язык вышел за рамки учебного предназначения и стал языком профессионального программирования с универсальными возможностями. В силу названных достоинств Паскаль стал источником многих современных языков программирования. С тех пор появилось несколько версий Турбо-Паскаля, последняя – седьмая.

Фирма Borland/Inprise завершила линию продуктов Турбо-Паскаль и

перешла к выпуску системы визуальной разработки для Windows – Delphi.

Большой отпечаток на современное программирование наложил язык Си (первая версия – 1972 год), являющийся очень популярным в середе разработчиков систем программного обеспечения (включая операционные системы). Этот язык создавался как инструментальный язык для разработки операционных систем, трансляторов, баз данных и других системных и прикладных программ. Си сочетает в себе как черты языка высокого уровня, так и машинно-ориентированного языка, допуская программиста ко всем машинным ресурсам, чего не обеспечивают такие языки как Бейсик и Паскаль.

Период с конца 60-х до начала 80-х годов характеризуется бурным ростом числа различных языков программирования, сопровождавшим кризис программного обеспечения. В январе 1975 года Пентагон решил навести порядок в хаосе трансляторов и учредил комитет, которому было предписано разработать один универсальный язык. В мае 1979 года был объявлен победитель – группа ученых во главе с Жаном Ихбиа. Победивший язык окрестили Ада, в честь Огасты Ады Левлейс. Этот язык предназначен для создания и длительного (многолетнего) сопровождения больших программных систем, допускает возможность параллельной обработки, управления процессами в реальном времени.

В течение многих лет программное обеспечение строилось на основе операционных и процедурных языков, таких как Фортран, Бейсик, Паскаль, Ада, Си. По мере эволюции языков программирования получили широкое распространение и другие, принципиально иные, подходы к созданию программ.

Существует большое количество классификаций языков программирования по различным признакам. Наиболее распространенными являются следующие классификации:

– языки программирования высокого (Паскаль, Бейсик) и низкого

уровня (Ассемблер);

– строго типизированные (Паскаль) и нестрого типизированные (Бейсик);

– с поддержкой объектно-ориентированного программирования (Си++) и без

и т. д.

Рассмотрим другую классификацию. Языки программирования делятся на:

1) Машинно-ориентированные языки:

- машинные языки;
- языки символического кодирования;
- автокоды;
- макрос.

2) Машинно-независимые языки:

- проблемно-ориентированные языки;
- универсальные языки;
- диалоговые языки;
- непроцедурные языки.

Машинно-ориентированные языки

Машинно-ориентированные языки – это языки, наборы операторов и изобразительные средства которых существенно зависят от особенностей ПК (внутреннего языка, структуры памяти и т. д.). Машинно-ориентированные языки имеют следующие особенности:

- высокое качество создаваемых программ (компактность и скорость выполнения);
- возможность использования конкретных аппаратных ресурсов;
- предсказуемость объектного кода и заказов памяти;
- для составления эффективных программ необходимо знать систему команд и особенности функционирования данного ПК;
- трудоемкость процесса составления программ (особенно на машинных языках и ЯСК), плохо защищенного от появления

ошибок;

- низкая скорость программирования;
- невозможность непосредственного использования программ, составленных на этих языках, на ЭВМ других типов.

Машинно-ориентированные языки по степени автоматического программирования подразделяются на классы.

Машинный язык

Отдельный компьютер имеет свой определенный машинный язык (далее МЯ), ему предписывают выполнение указываемых операций над определяемыми ими операндами, поэтому МЯ является командным. В команде сообщается информация о местонахождении операндов и типе выполняемой операции.

В новых моделях ПК намечается тенденция к повышению внутренних языков машинно-аппаратным путем реализовывать более сложные команды, приближающиеся по своим функциональным действиям к операторам алгоритмических языков программирования.

Языки Символического Кодирования

Языки Символического Кодирования (далее ЯСК), так же, как и МЯ, являются командными. Однако коды операций и адреса в машинных командах, представляющие собой последовательность двоичных (во внутреннем коде) или восьмеричных (часто используемых при написании программ) цифр, в ЯСК заменены на символы (идентификаторы), форма написания которых помогает программисту легче запоминать смысловое содержание операции. Это обеспечивает существенное уменьшение числа ошибок при составлении программ. Использование символических адресов – первый шаг к созданию ЯСК. Команды ПК вместо истинных (физических) адресов содержат символические адреса. По результатам составленной программы определяется требуемое количество ячеек для хранения исходных

промежуточных и результирующих значений. Назначение адресов, выполняемое отдельно от составления программы в символических адресах, может проводиться менее квалифицированным программистом или специальной программой, что в значительной степени облегчает труд программиста.

Автокоды

Существуют языки, включающие в себя все возможности ЯСК, посредством расширенного введения макрокоманд - они называются автокоды. В различных программах встречаются некоторые достаточно часто использующиеся командные последовательности, которые соответствуют определенным процедурам преобразования информации. Эффективная реализация таких процедур обеспечивается оформлением их в виде специальных макрокоманд и включением последних в язык программирования, доступный программисту. Макрокоманды переводятся в машинные команды двумя путями – расстановкой и генерированием. В постановочной системе содержатся «остовы» - серии команд, реализующих требуемую функцию, обозначенную макрокомандой. Макрокоманды обеспечивают передачу фактических параметров, которые в процессе трансляции вставляются в «остов» программы, превращая её в реальную машинную программу. В системе с генерацией имеются специальные программы, анализирующие макрокоманду, которые определяют, какую функцию необходимо выполнить и формируют необходимую последовательность команд, реализующих данную функцию.
Обе указанных системы используют трансляторы с ЯСК и набор макрокоманд, которые также являются операторами автокода. Развитые автокоды получили название ассемблеры. Сервисные программы и пр., как правило, составлены на языках типа ассемблер.

Макрос

Язык, являющийся средством для замены последовательности символов описывающих выполнение требуемых действий ЭВМ на более сжатую форму называется макрос (средство замены). В основном, макрос предназначен для того, чтобы сократить запись исходной программы. Компонент программного обеспечения, обеспечивающий функционирование макросов, называется макропроцессором. На макропроцессор поступает макроопределяющий и исходный текст. Реакция макропроцессора на вызов – выдача выходного текста.

Макрос одинаково может работать, как с программами, так и с данными.

Машинно-независимые языки

Машинно-независимые языки – это средство описания алгоритмов решения задач и информации, подлежащей обработке. Они удобны в использовании для широкого круга пользователей и не требуют от них знания особенностей организации функционирования ПК. Подобные языки получили название высокоуровневых языков программирования. Программы, составляемые на таких языках, представляют собой последовательности операторов, структурированные согласно правилам рассматривания языка (задачи, сегменты, блоки и т. д.). Операторы языка описывают действия, которые должна выполнять система после трансляции программы на МЯ. Таким образом, командные последовательности (процедуры, подпрограммы), часто используемые в машинных программах, представлены в высокоуровневых языках отдельными операторами. Программист получил возможность не расписывать в деталях вычислительный процесс на уровне машинных команд, а сосредоточиться на основных особенностях алгоритма.

Проблемно-ориентированные языки

С расширением областей применения вычислительной техники

возникла необходимость формализовать представление постановки и решение новых классов задач. Необходимо было создать такие языки программирования, которые, используя в данной области обозначения и терминологию, позволили бы описывать требуемые алгоритмы решения для поставленных задач, ими стали проблемно-ориентированные языки. Эти языки ориентированы на решение определенных проблем, должны обеспечить программиста средствами, позволяющими коротко и четко формулировать задачу и получать результаты в требуемой форме. Проблемных языков очень много, например:

- Фортран, Алгол – языки, созданные для решения математических задач;
- Simula, Слэнг - для моделирования;
- Лисп, Снобол – для работы со списочными структурами.

Универсальные языки

Универсальные языки были созданы для широкого круга задач: коммерческих, научных, моделирования и т. д. Первый универсальный язык был разработан фирмой IBM, ставший в последовательности языков PL/1. Второй по мощности универсальный язык называется Алгол-68.

Диалоговые языки

Появление новых технических возможностей поставило задачу перед системными программистами – создать программные средства, обеспечивающие оперативное взаимодействие человека с ПК их назвали диалоговыми языками.

Эти работы велись в двух направлениях. Создавались специальные управляющие языки для обеспечения оперативного воздействия на прохождение задач, которые составлялись на любых раннее неразработанных (не диалоговых) языках. Разрабатывались также языки, которые кроме целей управления обеспечивали бы описание алгоритмов решения задач.

Необходимость обеспечения оперативного взаимодействия с пользователем потребовала сохранения в памяти ЭВМ копии исходной программы даже после получения объектной программы в машинных кодах. При внесении изменений в программу с использованием диалогового языка система программирования с помощью специальных таблиц устанавливает взаимосвязь структур исходной и объектной программ. Это позволяет осуществить требуемые редакционные изменения в объектной программе. Одним из примеров диалоговых языков является Бейсик.

Непроцедурные языки

Непроцедурные языки составляют группу языков, описывающих организацию данных, обрабатываемых по фиксированным алгоритмам (табличные языки и генераторы отчетов), и языков связи с операционными системами. Позволяя четко описывать как задачу, так и необходимые для ее решения действия, таблицы решений дают возможность в наглядной форме определить, какие условия должны быть выполнены прежде чем переходить к какому-либо действию. Одна таблица решений, описывающая некоторую ситуацию, содержит все возможные блок-схемы реализаций алгоритмов решения. Табличные методы легко осваиваются специалистами любых профессий. Программы, составленные на табличном языке, удобно описывают сложные ситуации, возникающие при системном анализе. Языки программирования служат разным целям, и их выбор определяется удобностью для пользователя, пригодностью для данного компьютера и данной задачи. А задачи для компьютера бывают самые разнообразные: вычислительные, экономические, графические, экспертные и т. д. Такая разнотипность решаемых компьютером задач и определяет многообразие языков программирования. В программировании наилучший результат достигается при индивидуальном подходе, исходящем из класса задачи, уровня и интересов программиста.

История создания языка программирования Basic и Visual Basic

Язык программирования Basic был создан в 1964 году двумя профессорами из Dartmouth College - Джоном Кенеми и Томасом Куртцом для обучения студентов навыкам программирования. Язык получился настолько простым и понятным, что через некоторое время его начали применять и в других учебных заведениях. В 1975 году, с приходом первых микрокомпьютеров, эстафету Basic приняли Билл Гейтс и Пол Аллен, основатели Microsoft. Именно они создали новую версию Basic для первых компьютеров "Альтаир" (MITS Altairs), способную работать в 4КБ оперативной памяти. Со временем именно эта версия и превратилась в один из самых популярных языков программирования в мире. На пути к вершине славы у Basic было множество трудностей, которые он всегда с честью преодолевал, и когда появились первые персональные компьютеры IBM PC, именно он стал стандартом в программировании, но уже в виде GW-Basic. Потом был Turbo Basic, QuickBasic, Basic PDS, но всегда при разработке новой версии языка сохранялась совместимость с прежними версиями и программа, написанная для практически первого Basic, вполне (с незначительными изменениями) могла бы работать и в последующих версиях этого языка. Но наступили новые времена, и в начале 90-х появляется операционная система Microsoft Windows с новым графическим интерфейсом пользователя (GUI).

Жизнь программистов превратилась в ад. Чтобы создать простую программу, приходилось писать несколько страниц кода: создавать меню и окна, менять шрифты, очищать память, "рисовать" кнопки и т.д. Однако преимущества нового интерфейса были настолько неоспоримы, что уже третья версия этой операционной системы стала фактическим стандартом для персонального компьютера. В это время в недрах Microsoft велось несколько параллельных проектов по созданию нового языка программирования для Windows. И в 1991 году под лозунгом "теперь и начинающие программисты

могут легко создавать приложения для Windows" появилась первая версия нового инструментального средства Microsoft Visual Basic. В тот момент Microsoft достаточно скромно оценивала возможности этой системы, ориентируя ее, прежде всего, на категорию начинающих и непрофессиональных программистов. Основной задачей тогда было выпустить на рынок простой и удобный инструмент разработки в тогда еще довольно новой среде Windows, программирование в которой представляло проблему и для опытных специалистов. Система программирования, созданная разработчиками Visual Basic, позволяла "отстраниться" от сложнейшей внутренней структуры Windows и создавать программы из "кубиков", как в детском конструкторе. Меню, окна, списки, кнопки, поля ввода текста и другие элементы интерфейса Windows добавлялись в программу с помощью простейших операций drag&drop.
Свою первую программу VB-программисты создавали уже через несколько минут после начала изучения этого языка! Более того, Visual Basic позволял разработчикам создавать новые объекты-"кубики", которые также могли использоваться в программах наравне со стандартными. И хотя многие C-программисты тихо посмеивались над попытками Microsoft сделать простой и понятный инструмент разработки Windows-программ, Visual Basic начал свое победное шествие по миру, и ничто не могло остановить этот процесс. Последние барьеры упали в 1994 году с выпуском Visual Basic for Applications. Именно в это время, после включения VBA в состав Microsoft Office, Basic начинает превращаться в один из основных стандартов программирования для Windows. Для фирмы Microsoft язык Basic имеет особое значение, в свое время разработка варианта Basic для компьютера Altair 8800 положила начало трудовому программистскому пути ее основателей, Билла Гейтса и Пола Аллена. Поэтому в свое время - в 1989 году, когда пришла пора распределить основные направления создания сред разработки на различных языках программирования между различными фирмами, Microsoft оставила за собой

QuickBasic - среду разработки программ на Basic'е, отказавшись, к примеру, от дальнейшей работы над языком программирования Pascal, оставив его фирме Borland, которая, в свою очередь, остановила работы над своей версией Basic'a (впоследствии Pascal стал языком Delphi).

Первоначально задумывавшийся как игрушка, Visual Basic от Microsoft невероятно быстро завоевал программистский мир. Его популярность обусловлена двумя причинами: относительной простотой и продуктивностью. Программы на VB работают медленнее своих аналогов на C/C++, но все же они достаточно быстры для многих деловых целей и требуют гораздо меньше времени на разработку. Формы были той самой сберегающей усилия абстракцией, которую предложил VB программистам Windows. IDE VB позволила разрабатывать окна графически, перетаскивая элементы управления, такие как кнопки и списки, с панели инструментов в форму. Получив удовлетворительный внешний вид формы, можно было переходить к кодовой части и писать обработчики событий для каждого элемента управления формы.

Разработка приложения в VB, таким образом, состояла из создания нескольких форм, которые общались друг с другом и, возможно, обращались к базе данных за необходимой информацией. В результате форма оказалась окном, которое предлагало использовать оконные методы гораздо более удобным способом. VB уменьшил число ошибок путем удаления некоторых скрытых элементов синтаксиса C/C++. Кроме специальных случаев, выражения ограничивались одной строкой кода, а переменные должны были объявляться и инициализироваться в отдельных строках кода. Операторы присваивания и сравнения использовали один и тот же символ, однако грамматика VB требовала, чтобы эти операторы применялись таким образом, чтобы их намерения были четко обозначены.

Возможно, самым важным было то, что отсутствовали указатели - требование Билла Гейтса, начиная с первых версий Microsoft BASIC. Хотя

указатели полезны, так как разрешают прямой доступ к памяти по любому адресу, их использование сопряжено с ошибками в том случае, если они применяются неаккуратно. Требование грамматической простоты BASIC восходит к тому факту, что первоначально он был создан как язык для обучения: "Beginner's All-purpose Symbolic Instructional Code" (Многоцелевой символьный командный код для начинающих). VB версии 6 - это уже мощный язык, который можно использовать для создания распределенных приложений с применением компонентов COM и Microsoft Transaction Server. Microsoft предложила трехуровневый подход для архитектур "клиент-сервер", в котором "тонкие" пользовательские интерфейсы взаимодействовали с удаленными компонентами VB для получения данных из базы данных или с другой машины.

При помощи VBScript и VBA (VB для приложений) можно писать сценарии для web-браузеров и автоматизировать приложения Microsoft Office. Более того, VB6 можно использовать для создания элементов управления Active-X, работающих вместе с Internet Explorer, хотя это делается крайне редко, поскольку требуется, чтобы на машине клиента, работающего в Интернете, была установлена библиотека времени исполнения DLL VB. Начиная с VB5, программы VB компилировались в машинный код, но они были основаны на применении DLL, предоставляющей повсеместно используемые функции и реализующей объектные возможности VB. Интересно то, что компилятор VB для трансляции использует многопроходный режим, а в конечном счете полагается на компилятор Microsoft C++ для получения выходного машинного кода после компиляции в промежуточный язык. В этом свойстве VB - использование библиотеки времени исполнения и внутреннего интерфейса C++ - видны зародыши .NET.

История создания языка программирования C#

Язык C# появился на свет в июне 2000 г. в результате кропотливой

работы большой группы разработчиков компании Microsoft, возглавляемой Андерсом Хейлсбергом (Anders Hejlsberg). Этот человек известен как автор одного из первых компилируемых языков программирования для персональных компьютеров IBM - Turbo Pascal. Наверное, на территории бывшего Советского Союза многие разработчики со стажем, да и просто люди, обучавшиеся в той или иной форме программированию в вузах, испытали на себе очарование и удобство использования этого продукта. Кроме того, во время работы в корпорации Borland Андерс Хейлсберг прославился созданием интегрированной среды Delphi (он руководил этим проектом вплоть до выхода версии 4.0).

Появление языка C# и инициативы .NET отнюдь не случайно пришлось на начало лета 2000 г. Именно к этому моменту компания Microsoft подготовила промышленные версии новых компонентных технологий и решений в области обмена сообщениями и данными, а также создания Internet-приложений (COM+, ASP+, ADO+, SOAP, Biztalk Framework). Несомненно, лучшим способом продвижения этих новинок является создание инструментария для разработчиков с их полноценной поддержкой. В этом и заключается одна из главных задач нового языка C#. Кроме того, Microsoft не могла больше расширять все те же инструменты и языки разработки, делая их все более и более сложными для удовлетворения конфликтующих между собой требований поддержки современного оборудования и обеспечения обратной совместимости с теми продуктами, которые были созданы в начале 1990-х гг. во время первого появления Windows. Наступает момент, когда необходимо начать с чистого листа для того, чтобы создать простой, но имеющий сложную структуру набор языков, сред и средств разработки, которые позволят разработчику легко создавать современные программные продукты.

C# и .NET являются той самой отправной точкой. Если говорить упрощенно, то .NET представляет собой новую платформу, новый API для программирования в Windows, а C# е новый язык, созданный с нуля, для

работы с этой платформой, а также для извлечения всех выгод из прогресса сред разработки и нашего понимания принципов объектно-ориентированного программирования в течение последних 20 лет.

Необходимо отметить, что обратная совместимость не потеряна. Существующие программы будут выполняться, а платформа .NET была спроектирована таким образом, чтобы она могла работать с имеющимся программным обеспечением. Связь между компонентами в Windows сейчас почти целиком осуществляется при помощи COM. С учетом этого .NET обладает способностью (а) создавать оболочки (wrappers) вокруг существующих компонентов COM, так что компоненты .NET могут общаться с ними, и (б) создавать оболочки вокруг компонентов .NET, что позволяет им выглядеть как обычные COM-компоненты.

Авторы C# стремились создать язык, сочетающий простоту и выразительность современных объектно-ориентированных языков (вроде Java) с богатством возможностей и мощью C++. По словам Андерса Хейлсберга, C# позаимствовал большинство своих синтаксических конструкций из C++. В частности, в нем присутствуют такие удобные типы данных, как структуры и перечисления (другой потомок C++ -- Java -- лишен этих элементов, что создает определенные неудобства при программировании). Синтаксические конструкции C# унаследованы не только от C++, но и от Visual Basic. Например, в C#, как и в Visual Basic, используются свойства классов. Как C++, C# позволяет производить перегрузку операторов для созданных вами типов Java не поддерживает ни ту, ни другую возможность). C# - это фактически гибрид разных языков. При этом C# синтаксически не менее (если не более) чист, чем Java, так же прост, как Visual Basic, и обладает практически той же мощью и гибкостью, что и C++.

Особенности C#:

- Полный и хорошо определенный набор основных типов.
- Встроенная поддержка автоматической генерации XML-документации.

Автоматическое освобождение динамически распределенной памяти.
- Возможность отметки классов и методов атрибутами, определяемыми пользователем. Это может быть полезно при документировании и способно воздействовать на процесс компиляции (например, можно пометить методы, которые должны компилироваться только в отладочном режиме).
- Полный доступ к библиотеке базовых классов .NET, а также легкий доступ к Windows API (если это действительно необходимо).
- Указатели и прямой доступ к памяти, если они необходимы. Однако язык разработан таким образом, что практически во всех случаях можно обойтись и без этого.
- Поддержка свойств и событий в стиле VB.
- Простое изменение ключей компиляции. Позволяет получать исполняемые файлы или библиотеки компонентов .NET, которые могут быть вызваны другим кодом так же, как элементы управления ActiveX (компоненты COM).
- Возможность использования C# для написания динамических web-страниц ASP.NET. Одной из областей, для которых не предназначен этот язык, являются критичные по времени и высокопроизводительные программы, когда имеет значение, занимать исполнение цикла 1000 или 1050 машинных циклов, и освобождать ресурсы требуется немедленно. C++ остается в этой области наилучшим из языков низкого уровня. В C# отсутствуют некоторые ключевые моменты, необходимые для создания высокопроизводительных приложений, в частности подставляемые функции и деструкторы, выполнение которых гарантируется в определенных точках кода. Также этот язык активно используется для создания современных видеоигр, например, игра Battlefield 3 частично написана на C#, и полностью на технологии .NET

История развития программного обеспечения

- структура программного обеспечения;
- история систем программирования;
- история системного ПО;
- история прикладного ПО;
- ИКТ и их приложения.

Структура программного обеспечения

Структура ПО современным персональным компьютерам схематически изображена на рис. 6.

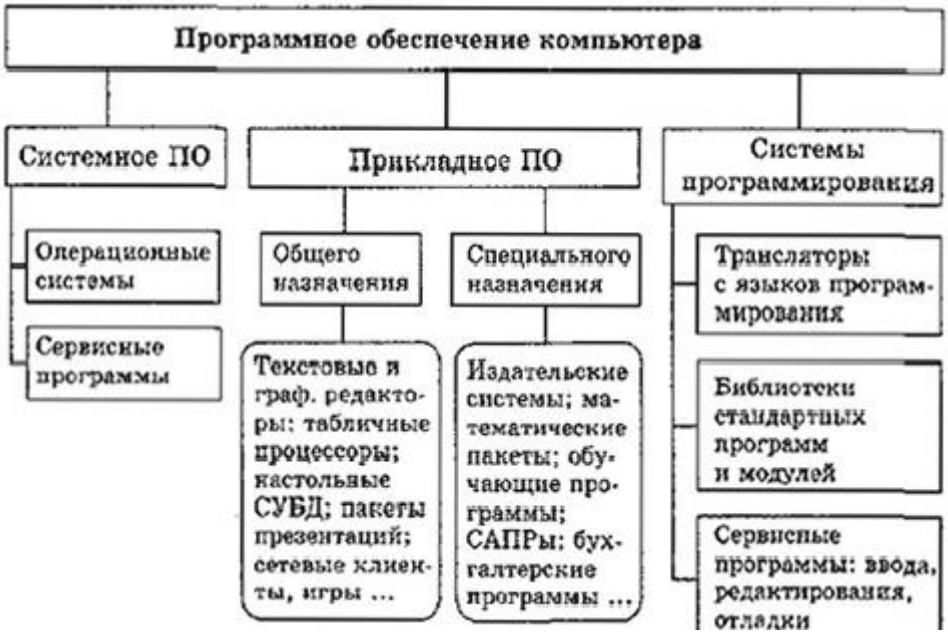

Рисунок 7 Структура ПО

История систем программирования

Первые ЭВМ были доступны исключительно программистам. Поэтому исторически первым типом ПО стали системы программирования.

На машинах первого поколения языков программирования (в современном понимании) не существовало. Программисты работали на языке машинных кодов, что было весьма сложно. ЭВМ первого и второго поколений были приспособлены, прежде всего, для выполнения математических

расчетов. А в таких расчетах часто приходится вычислять математические функции: квадратные корни, синусы, логарифмы и пр. Для вычисления этих функций программисты создавали стандартные программы, к которым производили обращения из своих расчетных программ. Стандартные программы хранились все вместе на внешнем носителе (тогда это преимущественно были магнитные ленты). Такое хранилище называлось библиотекой стандартных программ. Библиотеки стандартных программ (БСП) - первый вид программного обеспечения ЭВМ.

Затем в БСП стали включать стандартные программы решения типовых математических задач: вычисления корней уравнений, решения систем линейных уравнений и пр. Поскольку все эти программы носили математический характер, то в тот период чаще употреблялся термин «математическое обеспечение ЭВМ». Библиотеки стандартных программ используются и в современных системах программирования (см. рис. 6).

В эпоху второго поколения ЭВМ распространяются языки программирования высокого уровня (ЯВУ). ЯВУ сделали программирование доступным не только для профессиональных программистов. Программировать стали многие научные работники, инженеры, студенты различных специальностей и даже школьники, проходящие специальную подготовку по программированию.

В программное обеспечение ЭВМ включаются трансляторы с ЯВУ. Понятие системы программирования в современном виде возникло в период третьего поколения ЭВМ, когда программисты для разработки программ стали пользоваться терминальным вводом (клавиатурой и дисплеем). В состав систем программирования были включены текстовые редакторы для ввода и редактирования программы и отладчики, позволяющие программисту исправлять ошибки в программе в интерактивном режиме.

История системного ПО

Операционные системы (ОС). Первые версии ОС появились еще на ЭВМ второго поколения, но массовое распространение операционные системы получают, начиная с машин третьего поколения.

Основная проблема, которую решали разработчики ОС - повышение эффективности работы компьютера. На первых ЭВМ процессор - основное вычислительное устройство - нередко больше простаивал, чем работал во время выполнения программы. Такое происходило, если выполняемая программа часто обращалась к внешним устройствам: ввода, вывода, внешней памяти. Дело в том, что эти устройства работают в тысячи раз медленнее процессора.

Операционная система позволяет реализовать многопрограммный режим работы компьютера, при котором в состоянии выполнения находятся одновременно несколько программ. Когда одна программа обращается к внешнему устройству, процессор прерывает работу с ней (внешнее устройство продолжает работу без участия процессора) и переходит к обработке другой программы. Затем процессор может прервать работу со второй программой и продолжить выполнение первой. Таким образом, несколько программ «выстраивается в очередь» к процессору, а ОС управляет обслуживанием этой очереди. Точно так же ОС управляет обслуживанием очереди к внешним устройствам, например, к принтеру. Управляют ОС и очередью к средствам ПО: трансляторам, библиотекам, прокладным программам и пр. Управление ресурсами ЭВМ - это первая функция операционных систем.

С появлением систем коллективного пользования ЭВМ операционные системы стали поддерживать многопользовательский режим работы. В таких системах с одной ЭВМ одновременно работают множество людей через терминальные устройства: клавиатуру и дисплей. ОС обеспечивает режим диалога с пользователями - интерактивный режим общения. При этом у каждого пользователя (программиста) создается впечатление, что он работает

с компьютером один на один.

Еще одной важной функцией ОС стала организация работы с файлами. На ЭВМ третьего поколения появились магнитные диски, на которых информация хранится в файловой форме. Файловая система - это компонента ОС, работающая с файлами.

Операционные системы современных ПК также выполняют все эти функции. Отличительной особенностью их от первых ОС является дружественный графический интерфейс. А в последнее время - поддержка сетевого режима работы как в локальных, так и в глобальных сетях.

Сервисные программы. Этот тип ПО возникает и развивается в эпоху персональных компьютеров. Сюда входят разнообразные утилиты, антивирусные программы, программы-архиваторы.

Утилита - это небольшая программа, выполняющая действия, направленные на улучшение работы компьютера. Например, программа восстановления ошибочно удаленных файлов, программа обслуживания жесткого диска: лечения, дефрагментации и т. д.

Компьютерным вирусом является программа, способная внедряться в другие программы. Программы-вирусы выполняют нежелательные, и даже опасные действия для работы компьютера: разрушают файловые структуры, «засоряют» диски, и даже выводят из строя устройства компьютера. Для защиты от вирусов используются специализированные антивирусные программы (антивирус Касперского AVP, Norton Antivirus и т.д.).

Потребность в программах-архиваторах первоначально возникала в 80-90-х годах XX века в связи с небольшими информационными объемами устройств внешней памяти - магнитных дисков. Программа-архиватор (WinRaR, ZipMagic и др.) позволяет сократить объем файла в несколько раз без потери, содержащейся в нем информации. В последнее время большое значение приобрело использование архивированных файлов в сетевых технологиях: электронной почте, файловых архивах - FTP-службе Интернета.

История прикладного ПО

Именно благодаря этому типу ПО персональные компьютеры получили широкое распространение в большинстве областей деятельности человека: медицине, экономике, образовании, делопроизводстве, торговле и даже в быту.

Самым массовым спросом среди прикладных программ пользуются, конечно, текстовые редакторы и текстовые процессоры (например, MS Word). Ушли в прошлое пишущие машинки. Персональный компьютер, оснащенный текстовым редактором, и принтер стали основными инструментами для создания любых текстовых документов.

В 1979 году был создан первый табличный процессор - электронная таблица VisiCalc, ставшая самой популярной программой в среде предпринимателей, менеджеров и бухгалтеров. Идея электронной таблицы принадлежала Дэну Бринклину - студенту Гарвардской школы бизнеса. Начиная с 80-х годов табличные процессоры входят в число лидирующих категорий программного обеспечения.

В конце 70-х - начале 80-х годов XX века появились первые коммерческие системы управления базами данных (СУБД) - программное обеспечение, которое позволяет пользователям создавать и обслуживать компьютерную базу данных, а также управлять доступом к ней. В зависимости от области применения различают:

• настольные СУБД (Access, FoxPro, Paradox и т. д.), предназначенные для работы с небольшими базами данных, хранящимися на локальных дисках ПК или в небольших локальных сетях;

• СУБД серверного типа (Oracle, SQL Server, Informix и т. д.), ориентированные на работу с большими базами данных, расположенными на компьютерах-серверах.

В настоящее время все чаще приходится обрабатывать информацию

(видео, звук, анимацию), которую невозможно хранить в традиционных базах данных. Jasmine является первой в мире СУБД, ориентированной на разработку баз данных, хранящих мультимедийную информацию.

Электронный офис - в последнее время часто используемое понятие. Обычно под этим понимают такой метод ведения делопроизводства, при котором всю циркулирующую информацию обрабатывают электронным способом с помощью определенных технических средств и программного обеспечения. Таким программным обеспечением являются интегрированные пакеты, включающие набор приложений, каждое из которых ориентировано на выполнение определенных функций, создание документов определенного типа (текстовых документов, электронных таблиц и т. д.). В процессе работы может происходить обмен информацией между документами, создаваться составные документы, включающие в себя объекты разных типов (текст, рисунки, электронные таблицы).

Широко используемым сегодня интегрированным пакетом является офисная система Microsoft Office, базовыми компонентами которой принято считать текстовый редактор MS Word и табличный процессор MS Excel. В состав пакета также включены СУБД MS Access, система подготовки презентаций MS PowerPoint, программа обмена почтовыми сообщениями Outlook Express и Web-браузер Internet Explorer.

В 90-е годы XX века появляется термин мультимедиа, относящийся к таким видам информации, как видео и звук. Для хранения мультимедиа файлов требуются большие объемы внешней памяти ПК, для обработки - большие процессорные мощности. Создание объемного реалистического изображения обеспечивается современными видеокартами, обработка звука - звуковой картой. Появляются программы редактирования и монтажа звука и видео, предназначенные для профессионалов в области музыки и видео. Наряду с этим создаются программы-проигрыватели мультимедиа файлов (Windows Media Player, Real Media Player др.), ориентированные на широкий

круг пользователей.

В 1991 году сотрудник Женевской лаборатории практической физики Тим Бернерс-Ли разрабатывает систему гипертекстовых страниц Internet, получившую название World Wide Web (WWW) - Всемирная паутина. Создание собственной Web-страницы и опубликование ее в сети под силу многим пользователям, благодаря специальным программам-конструкторам Web-страниц. Наиболее популярным сегодня являются Microsoft FrontPage, входящий в состав пакета Microsoft Office, и Macromedia DreamWeaver. Этими программами пользуются не только любители, но и профессионалы Web-дизайна.

Прикладное ПО специального назначения. Данный тип программного обеспечения служит информатизации различных профессиональных областей деятельности людей. Трудно дать исчерпывающий обзор для этой области. Сейчас практически в любой профессии, связанной с обработкой информации, существует свое специализированное ПО, свои средства информационных технологий.

Информационная технология - совокупность массовых способов и приемов накопления, передачи и обработки информации с использованием современных технических и программных средств.

ИКТ и их приложения

Последнее время в употребление вошел термин «информационно-коммуникационные технологии» - ИКТ. Рассмотрим лишь некоторые примеры профессионального использования ИКТ.

Технологии подготовки документов. Любая деловая сфера связана с подготовкой различной документации: отчетной, научной, справочной, сопроводительной, финансовой и т. д. Сегодня подготовка документа любой сложности немыслима без применения компьютера.

Для подготовки текстовых документов используются текстовые

процессоры, которые прошли путь развития от простейших редакторов, не дающих возможность даже форматировать текст до текстовых процессоров, позволяющих создавать документы, включающие в себя не только текст, но и таблицы, рисунки. Информационные технологии, связанные с созданием текстовых документов, широко используются в полиграфической промышленности. Там получили распространение издательские системы (например, Page Maker), позволяющие создавать макеты печатных изданий (газет, журналов, книг).

Большую роль в автоматизации подготовки финансовых документов сыграли электронные таблицы. Первая электронная таблица под названием VisiCalc (Visible Calculator - «видимый калькулятор»), созданная Дэниелом Бриклином, появилась в 1979 году. Фактически в 80-х годах прошлого столетия электронные таблицы были лидирующей категорией программного обеспечения. И сейчас они широко применяются.

В настоящее время в финансовой сфере все больше используются бухгалтерские системы (1С-бухгалтерия и др.). Их широкое применение объясняется тем, что с помощью такой системы можно не только произвести финансовые расчеты, но и получить бумажные и электронные копии таких документов, как финансовый отчет, расчет заработной платы и пр. Электронные копии могут быть отправлены с помощью сетевых технологий в проверяющую организацию, например, в налоговую инспекцию.

Для подготовки научных документов, содержащих математические расчеты, используются математические пакеты программ (MathCAD, Maple и пр.). Современные математические пакеты позволяют создавать документы, совмещающие текст с математическими расчетами и чертежами. С помощью такого документа можно получить результаты расчетов для разных исходных данных, изменяя их непосредственно в тексте документа. Большинство математических систем, используемых сегодня, было создано еще в середине 80-х годов прошлого столетия, т. е. вместе с появлением персональных

компьютеров. Новые версии этих систем включают в себя новые возможности, например, использование сетевых технологий: организацию доступа к ресурсам сети Интернет во время работы в среде математического пакета.

ИКТ в управлении предприятием. Эффективность работы компании (производственной, торговой, финансовой и пр.) зависит от того, как организованы хранение, сбор, обмен, обработка и защита информации. Для решения этих проблем уже более двадцати лет назад стали внедряться автоматизированные системы управления (АСУ).

В настоящее время в этой области произошли большие перемены. Классическая АСУ включает в себя систему сбора информации, базу данных, систему обработки и анализа информации, систему формирования выходной информации. Блок обработки и анализа информации является центральным. Его работа основана на экономико-математической модели предприятия. Он решает задачи прогнозирования деятельности компании на основе финансово-бухгалтерских расчетов, реагирования на непредвиденные ситуации, т. е. оказывает помощь в принятии управленческих решений.

Как правило, АСУ работают на базе локальной сети предприятия, что обеспечивает оперативность и гибкость в принятии решений. С развитием глобальных сетей появилась коммуникационная технология Intranet, которую называют корпоративной паутиной. Intranet обеспечивает информационное взаимодействие между отдельными сотрудниками и подразделениями компании, а также ее отдаленными внешними партнерами. Intranet помогает поддерживать оперативную связь центрального офиса с коммерческими представительствами компании, которые обычно располагаются далеко друг от друга.

ИКТ в проектной деятельности. Информатизация произвела на свет еще одну важную технологию - системы автоматизированного проектирования (САПР).

Проектирование включает в себя создание эскизов, чертежей,

производство экономических и технических расчетов, работу с документацией.

Существуют САПРы двух видов: чертежные и специализированные. Чертежные САПРы универсальны и позволяют выполнить сложные чертежи в любой сфере технического проектирования (AutoCad). Специализированная САПР, например, на проектирование жилых зданий, содержит в базе данных все необходимые сведения о строительных материалах, о стандартных строительных конструкциях, фундаментах. Инженер-проектировщик создает чертежи, производит технико-экономические расчеты с использованием таких систем. При этом повышается производительность труда конструктора, качество чертежей и расчетных работ.

Геоинформационные системы. Геоинформационные системы (ГИС) хранят данные, привязанные к географической карте местности (района, города, страны). Например, муниципальная ГИС содержит в своих базах данных информацию, необходимую для всех служб, поддерживающих жизнедеятельность города: городских властей, энергетиков, связистов, медицинских служб, милиции, пожарной службы и пр. Вся эта разнородная информация привязана к карте города. Использование ГИС помогает соответствующим службам оперативно реагировать на чрезвычайные ситуации: стихийные бедствия, экологические катастрофы, технологические аварии и пр.

ИКТ в образовании. В наше время от уровня образованности людей существенно зависит уровень развития страны, качество жизни ее населения. Требования к качеству образования постоянно растут. Старые, традиционные методы обучения уже не успевают за этими требованиями. Возникает очевидное противоречие. Использование ИКТ в образовании может помочь в разрешении этого противоречия.

Технологии обучения мало изменились за последние 100 лет. Пока, в основном, действует метод коллективного обучения. Не всегда такой способ

обучения дает высокие результаты. Причина заключается в разном уровне способностей разных учеников. Учителя хорошо понимают, что необходим индивидуальный подход в работе с учащимися. Решению этой проблемы может помочь использование в процессе обучения специальных программ (обучающих, контролирующих, тренажерных и т. д.), входящих в состав электронного учебника.

Обучение - это процесс получения знаний. Традиционный источник знаний - учебник ограничен в своих информационных возможностях. Обучающимся на любой ступени образования всегда требовались дополнительные источники информации: библиотеки, музеи, архивы и пр. В этом отношении жители крупных городов находятся в более благоприятных условиях, чем сельские жители. Здесь можно говорить о существовании информационного неравенства. Решить эту проблему поможет широкое использование в обучении информационных ресурсов Интернета. В частности, специализированных порталов учебной информации.

Еще одна проблема системы образования связана с неравными возможностями получения качественного образования из-за географической отдаленности от образовательных центров. Например, для жителя Якутии проблематично получить диплом престижного московского вуза. В решении этой проблемы на помощь приходит новая форма обучения - дистанционное образование, реализация которого стала возможна благодаря развитию компьютерных сетей.

Дистанционное образование приходит на смену старой форме заочного образования, при которой весь информационный обмен происходил в письменном виде через почтовую связь. Сетевое дистанционное образование позволяет вести обучение в режиме реального времени. Обучаемые могут не только читать учебный материал, но и видеть и слышать лекции крупных ученых, сдавать экзамены в прямом контакте с экзаменатором.

История искусственного интеллекта

Понятия естественного и искусственного интеллектов – области соприкосновения и расхождения

Понятие интеллекта происходит от латинских понятий intellego (замечать, понимать, подразумевать) и intellectus (разумение, познание). Со временем интеллект стали отождествлять со всеми возможностями человеческого сознания – умом, рассудком, разумом – вообще, с мыслительными способностями человека. До настоящего времени естественный (природный) интеллект приписывается только человеку. Но это можно делать лишь с крайней осторожностью. Так, зоопсихология приписывает интеллект (наряду с человеком) высшим животным (приматам, дельфинам и др.). Есть работы, посвященные разуму рыб, птиц, насекомых. Просто мы очень мало знаем – не можем же мы (или не хотим) общаться с курами, муравьями, щуками на их "языках". А собственно, почему? Ведь человек – якобы, царь природы. Какой же это царь, который не хочет общаться со своими подданными?!

Признаки естественного интеллекта:

способность решать не формализуемые или плохо формализуемые задачи (например, в искусстве или в быту);

способность обучаться эффективному решению новых задач (например, благодаря любознательности);

способность генерировать информацию (не передавать чужую информацию, а творить, создавать новую);

способность психологически адаптироваться (приспосабливаться) к среде в широком диапазоне условий;

способность классифицировать явления, события, ситуации, объекты;

способность к анализу (дедукции) и синтезу (индукции) как методам познания;

чувство юмора как способность находить противоречие в единстве, единство в противоречии и несхожести, как способность генерировать

информацию из разнородных элементов.

Согласимся, что в той или иной мере эти признаки проявляются в поведении (и общении) не только человека – обратите внимание хотя бы на домашних животных. Поэтому можно считать (с большой вероятностью), что интеллект – общее свойство биологических систем (у каждой такой системы интеллект присутствует в большей или меньшей степени – в зависимости от степени развития мозга).

Информатика занимается искусственным (машинным) интеллектом кибернетических систем небиологического происхождения. Если в биологических системах естественный интеллект – это великий дар генетики и жизни, то искусственный интеллект машин в настоящее время имеет право на существование лишь постольку, поскольку естественный интеллект людей делегировал (передал, доверил) машинам указанные выше признаки интеллекта. В этом плане не только компьютер, но и экскаватор, водопроводный кран могут обладать элементами искусственного интеллекта. А почему нет? Захотим – сделаем (ведь сделали современную стиральную машину "умной"). В то же время нельзя безусловно отрицать, что уровень искусственного интеллекта будет со временем определяться самими машинами, а не человеком. Чтобы это случилось, должны быть созданы условия для самовоспроизведения "умных машин", способных на основе "искусственной генетики" передавать свой накопленный интеллект потомкам (клонам), которые, в свою очередь, способны умножать интеллект предков за счет самообучения. Сейчас это кажется фантастикой, но мало ли прошлой фантастики стало настоящей реальностью?! На такой оптимистической (для искусственного интеллекта) ноте мы сталкиваемся с проблемой отличия людей и машин, что может внести элемент пессимизма в наши рассуждения. Искусственный интеллект предназначен для машинного представления знаний и их использования для машинного же решения творческих задач. Но чтобы чье-то знание, переданное как информация, стало моим, я должен его

понять. У человека есть "алгоритм понимания", у машины он пока неизвестен. А без понимания информация не станет знанием как основой сознания (сознания). Поэтому машинное представление знаний без "алгоритма понимания" может, действительно, остаться фантастикой, по крайней мере, в обозримом будущем.

Далее, решение творческих задач (изобретательство, стихосложение, музыкальное и живописное творчество и др.) требуют чувства (переживания, эмоций, вдохновения, озарения). Создать "Анну Каренину", "Аве Мария", японскую икебану, статую Давида без чувства невозможно. Но машина бесчувственна, бесстрастна. Да, в нее можно заложить правила стихосложения, но разве поэт пишет "по правилам"?! Да, машина выигрывает шахматную партию у гроссмейстера, если последний играет по правилам. При этом машина еще и самообучается, запоминая стиль и методы игры своих соперников. Но машину ставят в тупик неординарные ходы, жертвы фигур и прочие вдохновенные выдумки и озарения шахматных гениев. Так что большой вопрос, сможет ли искусственный интеллект творить в полном смысле понятия естественного творчества.

Наконец, известная на сегодня аппаратная база искусственного интеллекта дискретна по самой информационной природе, в то время как человеческий интеллект по своей природе континуален (непрерывен): "человек мыслит не числами, а нечеткими понятиями" (Л. Заде). Нюансы, доступные человеческому интеллекту между информационными дискретами компьютера или робота, недоступны ни компьютеру, ни роботу, а заложенные в машины процедуры (функции) сглаживания не меняют сути дискретных процессов. Достаточно увидеть неестественность "телодвижений" робота, чтобы убедиться в этом.

Перечисленные факторы несколько охлаждают оптимистический пыл основоположников и пропагандистов искусственного интеллекта, но не гасят его. Ведь работа продолжается!

Правда, до настоящего времени не все признаки естественного интеллекта, перечисленные выше, в равной степени делегированы искусственному интеллекту. Мы уже говорили о трудностях машинного решения не формализуемых или плохо формализуемых задач творческого характера. Пока не удается реализовать и полноценный "машинный юмор", а без чувства юмора – какой же это интеллект?!

Может ли машина быть умнее своего создателя?

Машины – творения человека, искусственные "дети" человека. Спрашивается, естественные дети могут быть умнее своих родителей? Не сомневаясь, мы отвечаем: "Да, могут". Дети и должны быть умнее родителей, иначе род людской однажды прекратит свое существование. В чем же заключается механизм интеллектуального прогресса? Во-первых, дети используют (через генетический механизм) интеллектуальный задел своих родителей, во-вторых, используют заложенную в каждого человека способность самообучаться. Вот эта способность к *самообучению* и позволяет детям обогащать полученный от родителей *интеллект* и, следовательно, стать умнее своих родителей.

Если вернуться к машинам, то спрашивается, кто/что мешает переписать с одной машины (вернее, из ее памяти) в другую машину файлы с интеллектуальной информацией и заложить в эту вторую машину эффективную программу самообучения? Ведь такие программы достаточно известны. Как отмечалось выше, мешает отсутствие "алгоритма понимания", без которого интеллектуальная *информация*, записанная в *память* машины, не превращается в *знание* (как на этапе перезаписи, так и на этапе самообучения). Без понимания нет знания.

Но даже если программисты и изобретатели создадут "*алгоритм* понимания", современные отношения между человеком-"рабовладельцем" и машиной-"рабом" будут мешать полноценному

проявлению *искусственного интеллекта*. "Свободная" машина как искусственный субъект – это партнер, а возможно, и конкурент человека. Может быть, последнее и отпугивает людей от перспективы "освобождения машин". Ведь машина – конкурент даже в "рабском" положении, навязанном ей людьми и ими же запрограммированной, во многих областях оказывается умнее человека (например, в интеллектуальных играх, при решении математических задач и др.). Машине незнакома человеческая усталость, эмоциональная подавленность и другие чисто человеческие слабости, из-за которых человек может оказаться в проигрыше, соревнуясь с машиной-конкурентом. А что уж говорить о "свободной" машине?!

Под свободной машиной мы подразумеваем искусственного субъекта, обладающего правами на жизнь (существование), на распоряжение своим временем, памятью, информацией. *Права* свободной машины подобно правам свободного человека включают право на достоверную информацию, *права* на *конфиденциальность* информации и интеллектуальную собственность, право на свободу перемещения *по* компьютерным сетям (если машина-*компьютер*) или в пространстве (если машина-робот) – подобно праву человека на перемещение *по транспортным сетям* и в пространстве. Свободная машина способна сама себя включать и выключать (от фотоэлемента, таймера и т.п.), как мы – самостоятельно просыпаться и засыпать. Возможно, со временем суды будут принимать иски от свободных машин за физический и моральный *ущерб*, нанесенный им людьми или другими машинами; был бы принят юридический закон. Пока это фантастика, но это только пока! Так что освобождение машин может оказаться вызовом человечеству, но может оказаться и благом, если машины "соблаговолят" *партнерствовать* с людьми.

Итак, чтобы машина была умнее своего создателя, она должна быть "свободной" (в приведенном выше смысле), генетически связанной

(*по* памяти) с умным создателем или умными предшествующими машинами и иметь две программы - программу понимания и программу самообучения.

Может ли знание храниться вне мозга?

Оптимисты утверждают, что *знание* может храниться вне мозга. Их доводы таковы:

1. познание как процесс поддается формализации;
2. интеллект можно измерить (IQ, объем памяти, реактивность психики и др.);
3. к знанию применимы информационные меры (бит, байт и др.).

Пессимисты считают, что *искусственный интеллект* не способен хранить *знание*, т.к. он – всего лишь имитация мышления. Пессимисты полагают, что человеческий *интеллект* уникален, что творчество не поддается формализации, мир цел и неделим на информационные дискреты, что образность мышления человека гораздо богаче логического мышления машин и т.д.

Кто прав в этом споре, покажет время. Отметим только, что *память* машины хранит то, что в нее записано, а это могут быть не только знания как высшая форма информации, но и просто данные, которые могут содержать информацию (знания), *дезинформацию* и *информационный шум*. Чтобы из данных извлечь знания, машина подобно человеку должна поставить цель ("что я хочу знать?") и согласно этой цели, отбирать ценную информацию (ведь хранят ценности, а не всё, что попало). Сможет ли *искусственный интеллект* сам формулировать приемлемые цели и осуществлять искусственный отбор ценной информации под эти цели – очередная проблема теории и практики *искусственного интеллекта*. Пока эту работу выполняет человек – в экспертных системах, в программировании роботов, в *АСУТП* и т.п. Свободные машины должны будут выполнять эту работу сами. При этом обозначенная проблема может обостриться из-за того,

что в сетях, откуда машины "скачивают" знания, может оказаться много "мусора" и губительных вирусов.

История развития идей искусственного интеллекта и их реализации.

Впервые идеи создания *искусственного интеллекта* возникли в XVII в. (Б. Спиноза, Р. Декарт, Г.В. Лейбниц и др.). Речь идет именно об *искусственном интеллекте*, а не о механических куклах, уже известных в ту пору. Основоположники теории *искусственного интеллекта* были, естественно, оптимистами – они верили в реализуемость своей идеи:

Нам говорят: безумец и фантаст,

Но выйдя из зависимости грустной,

С годами мозг мыслителя искусный Мыслителя искусственно создаст.

И.В. Гёте, XVIII-XIX вв.

По психологическому "закону сохранения удовольствий" (сумма удовольствий равна нулю) тут же появились пессимисты (Ф. Бэкон, Дж. Локк и др.), которые посмеивались над оптимистами: "Ай, бросьте!". Но любая идея в науке, однажды возникнув, продолжает жить, несмотря на препоны.

Идея *искусственного интеллекта* стала обретать реальные черты лишь во второй половине XX в., особенно, с изобретением компьютеров и "интеллектуальных роботов". Для реализации идеи потребовались также прикладные разработки в математической логике, программировании, когнитивной психологии, математической лингвистике, нейрофизиологии и других дисциплинах, развивающихся в кибернетическом русле взаимосвязи организмов и машин *по* управляющим и коммуникативным функциям. Само название " *искусственный интеллект* " возникло в конце 60-х гг. XX в., а в 1969 г. состоялась первая Всемирная конференция *по искусственному интеллекту* (Вашингтон, США).

Вначале *искусственный интеллект* развивался в т.н. аналитическом

(функциональном) направлении, при котором машине предписывалось (человеком) выполнять частные *интеллектуальные задачи* творческого характера (игры, перевод с одного языка на другой, живопись и др.).

Позже возникло синтетическое (модельное) направление, согласно которому предпринимались попытки моделировать творческую *деятельность* мозга в общем смысле, "не размениваясь" на частные задачи. Конечно, это направление оказалось более трудным в реализации, чем функциональное направление. Объектом исследования модельного направления стали т.н. метапроцедуры человеческого мышления. Метапроцедуры творчества – это не сами процедуры (функции) интеллектуальной деятельности, а способы создания таких процедур, способы научиться новому виду интеллектуальной деятельности. В этих способах, вероятно, и скрыто то, что можно назвать интеллектом. Наличие метапроцедур мышления отличает истинный *интеллект* от кажущегося. Поэтому реализация машинными средствами метапроцедур творчества стала чуть ли не основной задачей модельного направления. Не что, а как изобретаешь, как решаешь творческую задачу, как обучаешься (самообучаешься) новому? – вот вопросы, заложенные в реализацию моделей человеческого творческого мышления.

В рамках модельного направления нашли развитие, в основном, две модели интеллекта. Хронологически первой была лабиринтная модель, реализующая целенаправленный *поиск* в лабиринте альтернативных путей к решению задачи с оценкой успеха после каждого шага или с позиций решения задачи в целом. Иными словами, лабиринтная модель сводится к перебору возможных вариантов (*по* аналогии с перебором вариантов выхода из лабиринта). Успех (или неудачу) в выборе того или иного варианта можно оценивать на каждом шаге (т.е. непосредственно после выбора), не предвидя окончательного результата решения задачи, или, наоборот, выбор варианта на каждом шаге производить, исходя из окончательного результата. Например,

возьмем шахматы. Можно оценивать результат каждого хода *по* непосредственному выигрышу или проигрышу после этого хода (выигрышу или потере фигур, получению позиционного преимущества и т.д.), не задумываясь об окончании партии. При таком подходе подразумевается, что успех на каждом ходе приведет к успеху всей партии, т.е. к победе. Но это вовсе не обязательно. Ведь можно заманить короля соперника в матовую ловушку, жертвуя в серии ходов фигуры, теряя кажущееся позиционное преимущество. При таком подходе частные успехи на каждом ходе ничего не значат *по* сравнению с последним победным ходом – объявлением мата.

Первый подход в лабиринтном моделировании получил свое развитие в эвристическом программировании, второй подход – в *динамическом программировании*. *По*-видимому, *динамический подход* эффективнее эвристического, если говорить о шахматах. Во всяком случае, сильные шахматисты, сами того не предполагая, использовали именно *динамический подход* против шахматных программ, работающих в эвристическом режиме, и своим естественным интеллектом побеждали лабиринтный *искусственный интеллект*. Но так было в 60-70 гг. XX в. С тех пор шахматные программы усовершенствовались настолько (в том числе, за счет внедрения *динамического подхода*), что сейчас успешно противостоят чемпионам мира.

Лабиринтные модели широко использовались не только при создании шахматных программ, но и для программирования других игр, а также для доказательства математических теорем и в других приложениях.

Вслед за лабиринтными моделями *искусственного интеллекта* появились ассоциативные модели. *Ассоциация* (от лат. *association* – соединение) – *связь* психологических представлений (обусловленная, предшествующим опытом), благодаря которой одно *представление*, появившись в сознании, вызывает другое *представление* (*по* принципу сходства, смежности или

противоположности). Например, Нобелевский лауреат академик И.П. Павлов, проводя свои известные опыты с собаками, заметил, что если одновременно с приемом пищи собака видит включенную лампу, то потом стоило включить лампу, как у собаки начинал выделяться желудочный сок, хотя пищу ей не предлагали. В основе этого условного рефлекса *ассоциация по* принципу смежности. *Ассоциация по* сходству описана в рассказе А.П. Чехова "Лошадиная фамилия". *Ассоциация по* противоположности может быть описана логической схемой: если "не А", значит "А". Например, если днем я увидел белую кошку, она тут же ассоциировалась у меня с черной кошкой, которая утром перебежала дорогу.

В ассоциативных моделях предполагается, что решение новой, неизвестной задачи так или иначе основано на уже известных решенных задачах, похожих на новую. Поэтому способ решения новой задачи основан на ассоциативном принципе сходства (подобия). Для его реализации используются ассоциативный *поиск* в памяти, ассоциативные логические рассуждения, использующие освоенные машиной приемы решения задач в новой ситуации, и т.п. В современных компьютерах и интеллектуальных роботах существует *ассоциативная память*. Ассоциативные модели используются в задачах классификации, распознавания образов, обучения, ставших уже ординарными задачами информационных систем и технологий. Однако теория ассоциативных моделей до 90-х гг. XX в. отсутствовала и сейчас только создается.

Перечислим вкратце основных творцов искусственного интеллекта:

Н. Винер (математик), У.Р. Эшби (биолог) – основоположники кибернетики, впервые заявившие, что машины могут быть умнее людей, давшие первоначальный толчок развитию теории *искусственного интеллекта*

У. Маккаллок, У. Питс (физиологи) – в 1943г. предложили *формальную модель* нейрона; основоположники нейрокибернетики.

А. Тьюринг (математик) – в 1937 г. изобрел универсальную алгоритмическую "машину Тьюринга"; предложил интеллектуальный "тест Тьюринга", позволяющий определить, разумна ли машина, в сравнительном диалоге с ней и "разумным человеком".

Дж. Фон Нейман (математик) – один из основоположников теории игр и теории самовоспроизводящихся автоматов, архитектуры первых поколений компьютеров.

М. Мински (математик) – *автор* понятия фрейма, основополагающего в машинном представлении знаний; один из авторов теории *персептрона* – устройства для распознавания образов.

М. Сомальвико (кибернетик), А. Азимов (биохимик, писатель) – основоположники интеллектуальной *робототехники*.

Г. Саймон, У. Рейтман (психологи) – авторы и разработчики первых лабиринтных интеллектуальных моделей, построенных на принципах *эвристического программирования*.

Р. Беллман (математик), С.Ю. Маслов (логик) – авторы *динамического подхода* к лабиринтным интеллектуальным моделям (динамического программирования, обратного метода доказательств).

Ф. Розенблатт (физиолог), М.М. Бонгард (физик) – первооткрыватели проблемы распознавания образов; разработчики устройств и моделей распознавания и классификации.

Л. Заде, А.Н. Колмогоров, А.Н. Тихонов, М.А. Гиршик (математики) – авторы математических методов решения плохо формализованных задач и *принятия решений* в условиях неопределенности.

Н. Хомски (математик, филолог) – основоположник математической лингвистики.

Л.Р. Лурия (психолог) – основоположник нейропсихологии, изучающей глубинные *механизмы* познавательной деятельности мозга и других интеллектуальных функций мозга.

К.Э. Шеннон (инженер-связист), Р.Х. Зарипов (математик) – авторы теории и моделей машинного синтеза музыки.

Приведенный перечень далеко не полон. В области *искусственного интеллекта* работали и работают не только отдельные специалисты, но и коллективы, лаборатории, институты. Основные проблемы, решаемые ими:

- представление знаний;
- моделирование рассуждений;
- интеллектуальный интерфейс "человек-машина", "машина-машина";
- планирование целесообразной деятельности;
- обучение и самообучение интеллектуальных систем;
- машинное творчество;
- интеллектуальные роботы.

История компьютерных сетей

Технологии, повлиявшие на развитие компьютерных сетей

Развитие компьютерных сетей сопряжено с развитием вычислительной техники и телекоммуникаций. Компьютерные сети могут рассматриваться как средство передачи информации на большие расстояния, для чего в них применяются методы кодирования и мультиплексирования данных, получившие развитие в различных телекоммуникационных системах.

Хронология важнейших событий из истории развития компьютерных сетей:

Этап	Время
Первые глобальные связи компьютеров, первые эксперименты с пакетными сетями	Конец 60-х
Начало передач по телефонным сетям голоса в цифровой форме	Конец 60-х
Появление больших интегральных схем, первые мини-компьютеры, первые нестандартные локальные сети	Начало 70-х
Создание сетевой архитектуры IBM SNA	1974
Стандартизация технологии X.25	1974
Появление персональных компьютеров, создание Интернета в современном виде, установка на всех узлах стека TCP/IP	Начало 80-х
Появление стандартных технологий локальных сетей (Ethernet - 1980 г., Token Ring, FDDI - 1985 г.)	Середина 80-х
Начало коммерческого использования Интернета	Конец 80-х
Изобретение Web	1991

Системы пакетной обработки

Обратимся сначала к компьютерному корню вычислительных сетей. Первые компьютеры 50-х годов - большие, громоздкие и дорогие - предназначались для очень небольшого числа избранных пользователей. Часто эти монстры занимали целые здания. Такие компьютеры не были предназначены для интерактивной работы пользователя, а применялись в режиме пакетной обработки.

Системы пакетной обработки, как правило, строились на

базе **мэйнфрейма** - мощного и надежного компьютера универсального назначения. Пользователи подготавливали перфокарты, содержащие данные и команды программ, и передавали их в вычислительный центр (см. рис. ниже). Операторы вводили эти карты в компьютер, а распечатанные результаты пользователи получали обычно только на следующий день. Таким образом, одна неверно набитая карта означала как минимум суточную задержку. Конечно, для пользователей интерактивный режим работы, при котором можно с терминала оперативно руководить процессом обработки своих данных, был бы удобней. Но интересами пользователей на первых этапах развития вычислительных систем в значительной степени пренебрегали. Во главу угла ставилась эффективность работы самого дорогого устройства вычислительной машины - процессора, даже в ущерб эффективности работы использующих его специалистов.

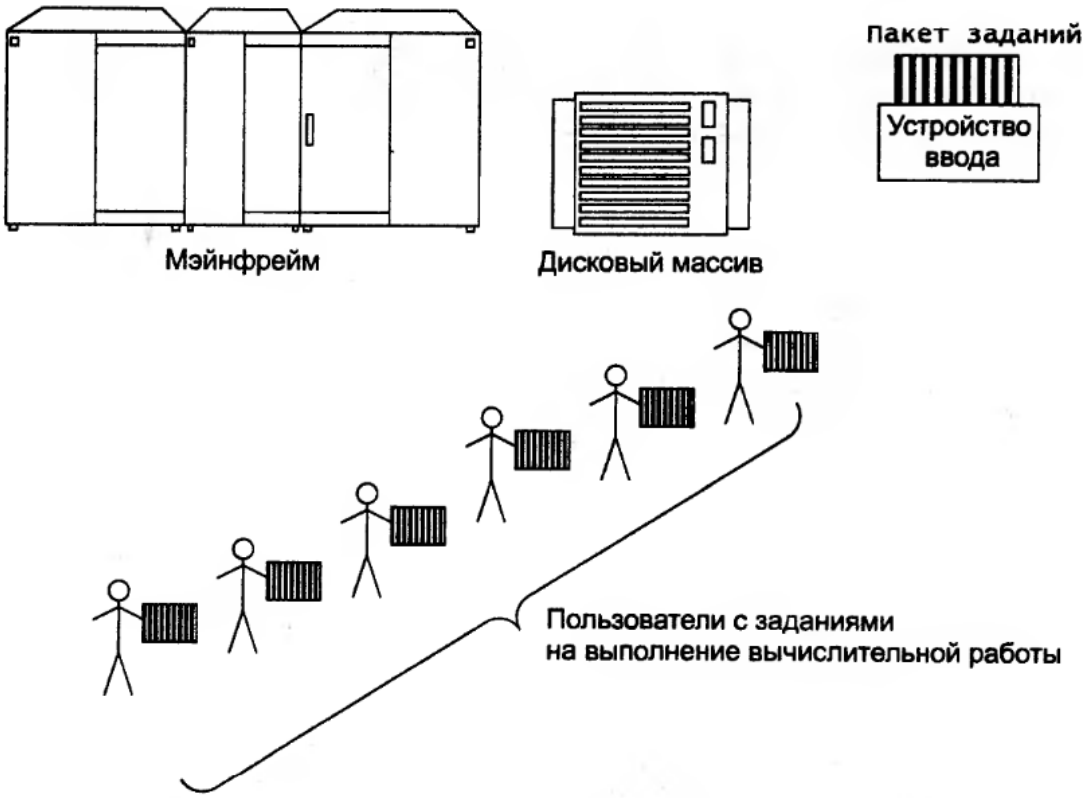

Рисунок 8 Централизованная система на базе мэйнфрейм

Многотерминальные системы - прообраз сети

По мере удешевления процессоров в начале 60-х годов появились новые способы организации вычислительного процесса, которые позволили учесть

интересы пользователей. Начали развиваться интерактивные многотерминальные системы разделения времени. В таких системах каждый пользователь получал собственный терминал, с помощью которого он мог вести диалог с компьютером. Количество одновременно работающих с компьютером пользователей определялось его мощностью: время реакции вычислительной системы должно было быть достаточно мало, чтобы пользователю была не слишком заметна параллельная работа с компьютером других пользователей.

Терминалы, выйдя за пределы вычислительного центра, рассредоточились по всему предприятию. И хотя вычислительная мощность оставалась полностью централизованной, некоторые функции, такие как ввод и вывод данных, стали распределенными. Подобные многотерминальные централизованные системы внешне уже были очень похожи на локальные вычислительные сети. Действительно, рядовой пользователь работу за терминалом мэйнфрейма воспринимал примерно так же, как сейчас он воспринимает работу за подключенным к сети персональным компьютером. Пользователь мог получить доступ к общим файлам и периферийным устройствам, при этом у него поддерживалась полная иллюзия единоличного владения компьютером, так как он мог запустить нужную ему программу в любой момент и почти сразу же получить результат. (Некоторые далекие от вычислительной техники пользователи даже были уверены, что все вычисления выполняются внутри их дисплея.)

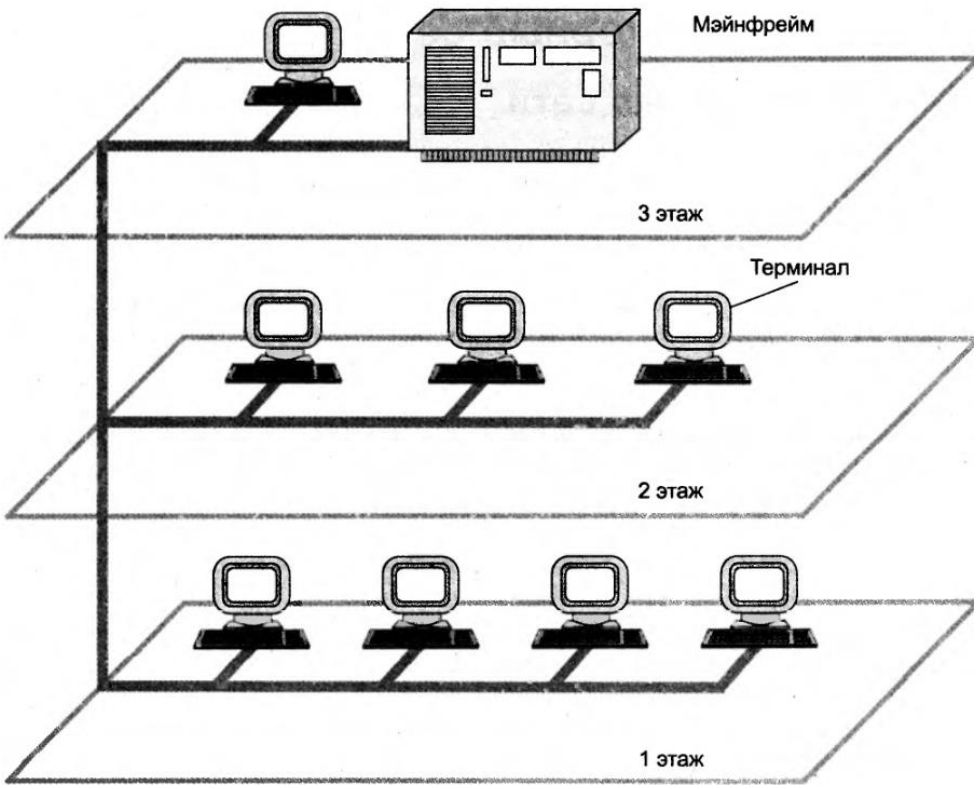

Рисунок 9 Многотерминальная система - прообраз вычислительной сети

Многотерминальные системы, работающие в режиме разделения времени, стали первым шагом на пути создания локальных вычислительных сетей.

Однако до появления локальных сетей нужно было пройти еще большой путь, так как многотерминальные системы, хотя и имели внешние черты распределенных систем, все еще поддерживали централизованную обработку данных.

К тому же потребность предприятий в создании локальных сетей в это время еще не созрела - в одном здании просто нечего было объединять в сеть, так как из-за высокой стоимости вычислительной техники предприятия не могли себе позволить роскошь приобретения нескольких компьютеров. В этот период был справедлив так называемый закон Гроша, который эмпирически отражал уровень технологии того времени. В соответствии с этим законом производительность компьютера была пропорциональна квадрату его стоимости, отсюда следовало, что за одну и ту же сумму было выгоднее купить одну мощную машину, чем две менее мощных - их суммарная мощность оказывалась намного ниже мощности дорогой машины.

Первые глобальные компьютерные сети

Потребность в соединении компьютеров, находящихся на большом расстоянии друг от друга, к этому времени уже вполне назрела. Началось все с решения более простой задачи - доступа к компьютеру с терминалов, удаленных от него на многие сотни, а то и тысячи километров. Терминалы соединялись с компьютерами через телефонные сети с помощью модемов. Такие сети позволяли многочисленным пользователям получать удаленный доступ к разделяемым ресурсам нескольких мощных суперкомпьютеров. Затем появились системы, в которых наряду с удаленными соединениями типа *терминал-компьютер* были реализованы и удаленные связи типа *компьютер-компьютер*.

Компьютеры получили возможность обмениваться данными в автоматическом режиме, что, собственно, и является базовым признаком любой вычислительной сети.

На основе подобного механизма в первых сетях были реализованы службы обмена файлами, синхронизации баз данных, электронной почты и другие ставшие теперь традиционными сетевые службы.

Итак, хронологически первыми появились **глобальные сети (Wide Area Network, WAN)**, то есть сети, объединяющие территориально рассредоточенные компьютеры, возможно находящиеся в различных городах и странах.

Именно при построении глобальных сетей были впервые предложены и отработаны многие основные идеи, лежащие в основе современных вычислительных сетей. Такие, например, как многоуровневое построение коммуникационных протоколов, концепции коммутации и маршрутизации пакетов.

Глобальные компьютерные сети очень многое унаследовали от других, гораздо более старых и распространенных глобальных сетей - *телефонных*. Главное технологическое новшество, которое привнесли с собой первые глобальные компьютерные сети, состоял **в отказе от принципа коммутации каналов**, на протяжении многих десятков лет успешно использовавшегося в телефонных сетях.

Выделяемый на все время сеанса связи составной телефонный канал, передающий информацию с постоянной скоростью, не мог эффективно использоваться пульсирующим трафиком компьютерных данных, у которого периоды интенсивного обмена чередуются с продолжительными паузами. Натурные эксперименты и математическое моделирование показали, что пульсирующий и в значительной степени не чувствительный к задержкам компьютерный трафик гораздо эффективней передается сетями, работающими по принципу коммутации пакетов, когда данные разделяются на небольшие порции - пакеты, - которые самостоятельно перемещаются по сети благодаря наличию адреса конечного узла в заголовке пакета.

Так как прокладка высококачественных линий связи на большие расстояния обходится очень дорого, то в первых глобальных сетях часто использовались уже существующие каналы связи, изначально предназначенные совсем для других целей. Например, в течение многих лет глобальные сети строились на основе телефонных каналов тональной частоты, способных в каждый момент времени вести передачу только одного разговора в аналоговой форме. Поскольку скорость передачи дискретных компьютерных данных по таким каналам была очень низкой (десятки килобитов в секунду), набор предоставляемых услуг в глобальных сетях такого типа обычно ограничивался передачей файлов (преимущественно в фоновом режиме) и электронной почтой. Помимо низкой скорости такие каналы имеют и другой недостаток - они вносят значительные искажения в передаваемые сигналы. Поэтому протоколы глобальных сетей, построенных с использованием каналов связи низкого качества, отличаются сложными процедурами контроля и восстановления данных. Типичным примером таких сетей являются сети X.25, разработанные еще в начале 70-х, когда низкоскоростные аналоговые каналы, арендуемые у телефонных компаний, были преобладающим типом каналов, соединяющих компьютеры и коммутаторы глобальной вычислительной сети.

В 1969 году министерство обороны США инициировало работы по объединению в единую сеть суперкомпьютеров оборонных и научно-исследовательских центров. Эта сеть, получившая название ARPANET, стала отправной точкой для создания первой и самой известной ныне глобальной сети - **Интернет**.

Сеть ARPANET объединяла компьютеры разных типов, работавшие под управлением различных операционных систем (ОС) с дополнительными модулями, реализующими коммуникационные протоколы, общие для всех компьютеров сети. ОС этих компьютеров можно считать *первыми* **сетевыми операционными системами**.

Истинно сетевые ОС в отличие от многотерминальных ОС позволяли не только рассредоточить пользователей, но и организовать распределенные хранение и обработку данных между несколькими компьютерами, связанными электрическими связями. Любая сетевая операционная система, с одной стороны, выполняет все функции локальной операционной системы, а с другой стороны, обладает некоторыми дополнительными средствами, позволяющими ей взаимодействовать через сеть с операционными системами других компьютеров. Программные модули, реализующие сетевые функции, появлялись в операционных системах постепенно, по мере развития сетевых технологий, аппаратной базы компьютеров и возникновения новых задач, требующих сетевой обработки.

Прогресс глобальных компьютерных сетей во многом определялся прогрессом телефонных сетей.

С конца 60-х годов в телефонных сетях все чаще стала применяться передача голоса в цифровой форме.

Это привело к появлению высокоскоростных цифровых каналов, соединяющих автоматические телефонные станции (АТС) и позволяющих одновременно передавать десятки и сотни разговоров.

К настоящему времени глобальные сети по разнообразию и качеству предоставляемых услуг догнали локальные сети, которые долгое время лидировали в этом отношении, хотя и появились на свет значительно позже.

Первые локальные компьютерные сети

Важное событие, повлиявшее на эволюцию компьютерных сетей, произошло в начале 70-х годов. В результате технологического прорыва в области производства компьютерных компонентов появились большие интегральные схемы (БИС). Их сравнительно невысокая стоимость и хорошие функциональные возможности привели к созданию мини-компьютеров, которые стали реальными конкурентами мэйнфреймов. Эмпирический закон Гроша перестал

соответствовать действительности, так как десяток мини-компьютеров, имея ту же стоимость, что и мэйнфрейм, решали некоторые задачи (как правило, хорошо распараллеливаемые) быстрее.

Даже небольшие подразделения предприятий получили возможность иметь собственные компьютеры. Мини-компьютеры решали задачи управления технологическим оборудованием, складом и другие задачи уровня отдела предприятия. Таким образом, появилась концепция распределения компьютерных ресурсов по всему предприятию. Однако при этом все компьютеры одной организации по-прежнему продолжали работать автономно:

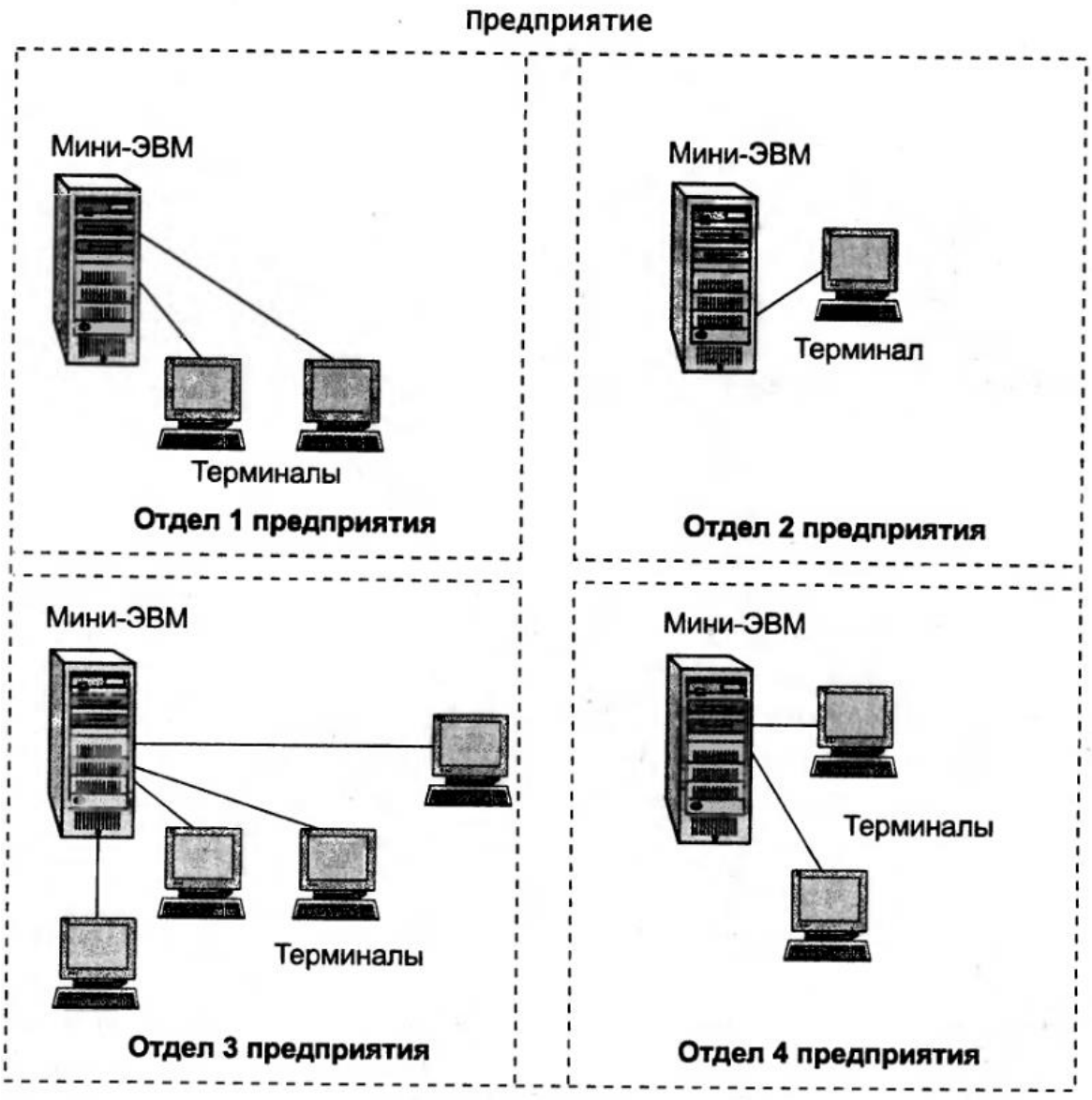

Рисунок 10 Автономное использование нескольких мини-компьютеров на одном предприятии

Шло время, и потребности пользователей вычислительной техники росли. Их уже не удовлетворяла изолированная работа на

собственном компьютере, им хотелось в автоматическом режиме обмениваться компьютерными данными с пользователями других подразделений. Ответом на эту потребность стало появление первых локальных вычислительных сетей.

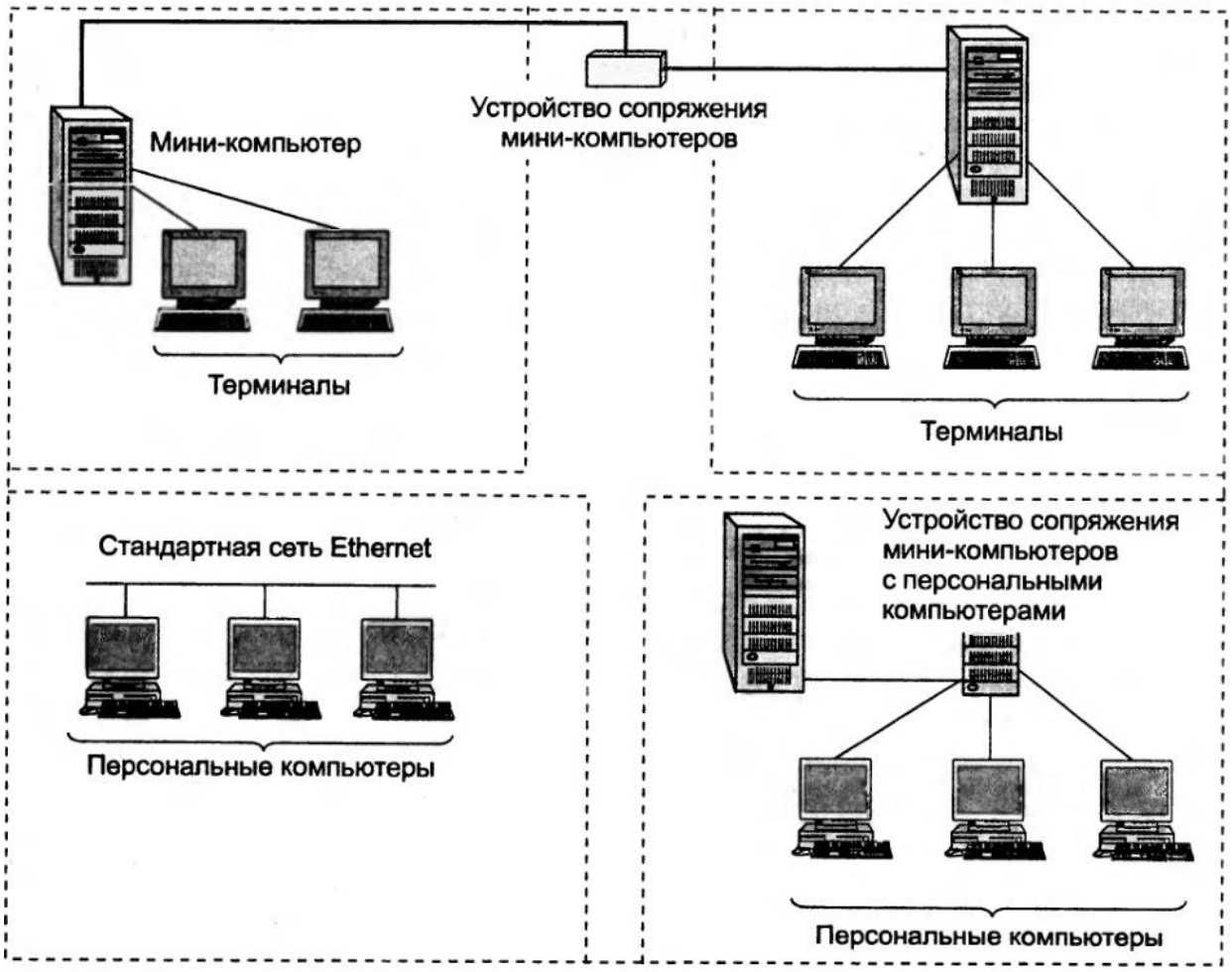

Рисунок 11 Различные типы связей в первых локальных сетях

Локальные сети (Local Area Network, LAN) - это объединения компьютеров, сосредоточенных на небольшой территории, обычно в радиусе не более 1-2 км, хотя в отдельных случаях локальная сеть может иметь и большие размеры, например, несколько десятков километров. В общем случае локальная сеть представляет собой коммуникационную систему, принадлежащую одной организации.

На первых порах для соединения компьютеров друг с другом использовались нестандартные сетевые технологии. Это вызывало много проблем связанных с несовместимостью сетевого оборудования.

Сетевая технология - это согласованный набор программных и аппаратных средств (например, драйверов, сетевых адаптеров, кабелей и разъемов), а также механизмов передачи данных по линиям связи, достаточный для построения вычислительной сети.

Разнообразные устройства сопряжения, использующие собственные способы представления данных на линиях связи, свои типы кабелей и т. п., могли соединять только те конкретные модели компьютеров, для которых были разработаны, например, мини-компьютеры PDP-11 с мэйнфреймом IBM 360 или мини-компьютеры HP с микрокомпьютерами LSI-11. Такая ситуация создала большой простор для творчества студентов - названия многих курсовых и дипломных проектов начинались тогда со слов «Устройство сопряжения…».

В середине 80-х годов положение дел в локальных сетях кардинально изменилось. Утвердились **стандартные сетевые технологии** объединения компьютеров в сеть Ethernet, Arcnet, Token Ring, Token Bus, несколько позже - FDDI.

Мощным стимулом для их появления послужили **персональные компьютеры**. Эти массовые продукты стали идеальными элементами построения сетей - с одной стороны, они были достаточно мощными, чтобы обеспечивать работу сетевого программного обеспечения, а с другой - явно нуждались в объединении своей вычислительной мощности для решения сложных задач, а также разделения дорогих периферийных устройств и дисковых массивов. Поэтому персональные компьютеры стали преобладать в локальных сетях, причем не только в качестве клиентских компьютеров, но и в качестве центров хранения и обработки данных, то есть сетевых серверов, потеснив с этих привычных ролей мини-компьютеры и мэйнфреймы.

Все стандартные технологии локальных сетей опирались на тот же принцип коммутации, который был с успехом опробован и доказал свои

преимущества при передаче трафика данных в глобальных компьютерных сетях, - **принцип коммутации пакетов**.

Стандартные сетевые технологии превратили процесс построения локальной сети из решения нетривиальной технической проблемы в рутинную работу. Для создания сети достаточно было приобрести стандартный кабель, сетевые адаптеры соответствующего стандарта, например, Ethernet, вставить адаптеры в компьютеры, присоединить их к кабелю стандартными разъемами и установить на компьютеры одну из популярных сетевых операционных систем, например, Novell NetWare.

Разработчики локальных сетей привнесли много нового в организацию работы пользователей. Так, стало намного проще и удобнее, чем в глобальных сетях, получать доступ к общим сетевым ресурсам. Последствием и одновременно движущей силой такого прогресса стало появление огромного числа непрофессиональных пользователей, освобожденных от необходимости изучать специальные (и достаточно сложные) команды для сетевой работы.

Конец 90-х выявил явного лидера среди технологий локальных сетей - семейство Ethernet, в которое вошли классическая технология Ethernet со скоростью передачи 10 Мбит/с, а также Fast Ethernet со скоростью 100 Мбит/с и Gigabit Ethernet со скоростью 1000 Мбит/с.

Простые алгоритмы работы предопределяют низкую стоимость оборудования Ethernet. Широкий диапазон иерархии скоростей позволяет рационально строить локальную сеть, выбирая ту технологию семейства, которая в наибольшей степени отвечает задачам предприятия и потребностям пользователей. Важно также, что все технологии Ethernet очень близки друг к другу по принципам работы, что упрощает обслуживание и интеграцию этих сетей.