Министерство образования и науки РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГОУВПО «АмГУ»

УТВЕРЖДАЮ
Зав. кафедрой ИУС
_____А.В. Бушманов
«___»_____

УЧЕБНО – МЕТОДИЧЕСКИЙ КОМПЛЕКС

по дисциплине «История и методология информатики и вычислитель ной техники»

для студентов направления подготовки 230100.68 – Информатика и вычислительная техника

Составитель: доцент, к.т.н. Самохвалова С.Г.

Факультет Математики и информатики

Кафедра информационных и управляющих систем

2009

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

С.Г. Самохвалова

Учебно-методический комплекс по дисциплине «История и методология информатики и вычислительной техники» для студентов очной формы обучения направления подготовки 230100.68 – Информатика и вычислительная техника. - Благовещенск: Амурский гос. ун-т, 2010. – с.

Учебно-методические рекомендации ориентированы на оказание помощи студентам очной формы обучения по направления подготовки 230100.68 – Информатика и вычислительная техника для успешного освоения дисциплины «История и методология информатики и вычислительной техники».

СОДЕРЖАНИЕ

Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ

Проректор по УР
_____ В.В. Проказин
«___» _____ 2009 г.

РАБОЧАЯ ПРОГРАММА

По дисциплине: История и методология информатики и вычислительной техники

По направлению подготовки: 230100.68 – Информатика и вычислительная техника

КУРС: 6 СЕМЕСТР: А

ЛЕКЦИИ: 18 (ЧАС.) ЭКЗАМЕН: А СЕМЕСТР

Лабораторные занятия: 36 (час.) Зачет: нет

Самостоятельная работа: 49 (час.)

Всего часов: 103 (час.)

Составитель: к.т.н.. доцент, Самохвалова С.Г.
Факультет Математики и информатики
Кафедра Информационных и управляющих систем

2009 г.

Рабочая программа составлена на основании Государственного образовательного стандарта ВПО подготовки магистра по направлению 230100.68 – Информатика и вычислительная техника

Рабочая программа обсуждена на заседании кафедры Информационных и управляющих систем

«___» _____ 2009 г., протокол № ___

Заведующий кафедрой А.В. Бушманов

Рабочая программа одобрена на заседании УМС по направлению подготовки 230100 – информатика и вычислительная техника

«___» _____ 2009 г., протокол № ___

Председатель В.В. Еремина

Согласовано
Начальник УМУ
_____ Г.Н. Торопчина
«___» _____ 2009 г.

Согласовано
Председатель УМС факультета
_____ С.Г. Самохвалова
«___» _____ 2009 г.

Согласовано
Заведующий выпускающей кафедрой
_____ А.В. Бушманов
«___» _____ 2009 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1. Цели и задачи изучения

1.1. Цель преподавания дисциплины состоит в:

- ознакомлении магистрантов с этапами исторического развития информатики и вычислительной техники;
- изучение новых парадигм построения вычислительных систем;
- освоение языковых средств для моделирования вычислений при различных способах построения вычислительных сред;
- развитии способностей исследования архитектурных решений в программировании.

1.2. В результате изучения курса магистрант должен знать:

- перспективы и тенденции развития информационных технологий;
- специальную научно-техническую литературу по тематике курса;
- современные информационные технологии, применяемые в научных исследованиях;
- программные продукты, применяемые в производственной сфере.

уметь:

- формулировать и решать задачи, возникающие в производственной и научно-исследовательской сфере для различных парадигм построения вычислительных средств;
- использовать современные методы, средства и технологии программирования при разработке систем;
- осуществлять сбор, обработку, анализ и систематизацию научно-технической информации, применять для этого современные информационные технологии.

иметь навыки:

- взаимодействия с коллегами и коллективом при исследовании и разработке компьютерных программ.

1. Содержание дисциплины

1.1. Федеральный компонент ДНМ.02

Требования государственного образовательного стандарта

“Докомпьютерная” информатика: алгоритмы и их анализ в математике, машинная обработка статистических данных, теория алгоритмов и

математическая логика; история и этапы эволюции вычислительной техники; кибернетика и информатика; компьютерная математика; численные методы и аналитические вычисления; развитие языков и технологии программирования; основные парадигмы программирования; эволюция проблем человеко-машинного взаимодействия и методов их решения; системы искусственного интеллекта; эволюция архитектуры вычислительных систем и сетей; компьютерная графика и системы мультимедиа; формирование информатики как фундаментальной науки.

1.2. Наименование тем, их содержание, объем в лекционных часах

ТЕМАТИЧЕСКИЙ ПЛАН ЛЕКЦИОННЫХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	Ведение	2
2	«Докомпьютерная» информатика	4
3	История и этапы эволюции вычислительной технике	4
4	Компьютерная математика	2
5	Развитие языков и технологий программирования	2
6	Эволюция проблем человеко-машинного взаимодействия и методов их решения	2
7	Эволюция архитектуры вычислительных систем и сетей	2
ИТОГО		**18**

Тема 1. Введение

Предмет и основные задачи истории и методологии информатики и вычислительной технике. Основные определения и понятия.

Тема 2. «Докомпьютерная» информатика

Алгоритмы и их анализ в математике, сложность алгоритмов, машинная обработка статистических данных, теория алгоритмов, математическая логика. Системы счисления. Абак и счеты. Логарифмическая линейка. Арифмометр. Вычислительные машины Бэббиджа (программное управление). Алгебра Буля. Табулятор Холлерита, счетно-перфорационные машины. Электромеханические и релейные машины. К. Цузе, проект MARK-1 Айкена. Аналоговые вычислительные машины.

Тема 3. История и этапы эволюции вычислительной техники

История и этапы эволюции вычислительной техники, поколения ЭВМ, развитие операционных систем и средств программирования, кибернетика и информатика. ENIAC, EDSAC, МЭСМ, М-1. Роль первых ученых - разработчиков компьютеров – Атанасова, Эккерта и Моучли, Дж. фон Неймана, С.А. Лебедева, И.С. Брука.

Поколения ЭВМ. Семейство машин IBM 360/370, машины «Атлас» фирмы ICL, машины фирм Burroughs, CDC, DEC. Отечественные ЭВМ серий

«Стрела», БЭСМ, М-20, «Урал», «Минск». ЭВМ «Сетунь». ЭВМ БЭСМ-6. Семейства ЕС ЭВМ, СМ ЭВМ и «Электроника». Отечественные ученые – разработчики ЭВМ – Ю.Я. Базилевский, В.А. Мельников, В.С. Бурцев, Б.И. Рамеев, В.В. Пржиялковский, Н.П. Брусенцов, М.А. Карцев, Б.Н. Наумов.

Тема 4. Компьютерная математика

Компьютерная математика: численные методы и аналитические вычисления. История математического моделирования и вычислительного эксперимента (Самарский А.А.). История систем массового обслуживания населения («Сирена», «Экспресс»).

Тема 5. Развитие языков и технологий программирования

Развитие языков и технологий программирования, основные парадигмы программирования. Первые языки – Фортран, Алгол-60, Кобол. Языки Ada, Pascal, PL/1. История развития объектно-ориентированного программирования. Simula и Smalltalk. Языки C и Java.

Тема 6. Эволюция проблем человеко-машинного взаимодействия и методов их решения

Развитие параллелизма в работе устройств компьютера, многопроцессорные и многомашинные вычислительные системы. Суперкомпьютеры. ILLIAC IV. Векторно - конвейерные ЭВМ. «Cray-1» и другие ЭВМ Сеймура Крея. Многопроцессорные ЭВМ классов SMP, MPP, NUMA. Вычислительные кластеры. СуперЭВМ в списке «TOP-500». Отечественные многопроцессорные вычислительные комплексы «Эльбрус-2» (Бурцев В.С.), ПС-2000 и ПС-3000 (Прангишвили И.В.), МВС-100, МВС-1000 и МВС-1000М (В.К. Левин).

Тема 7. Эволюция архитектуры вычислительных систем и сетей

Начальный период развития сетей. Сети с коммутацией каналов. Сети пакетной коммутации. От сети ARPAnet до Интернета. Локальные вычислительные сети. Сетевые протоколы. Сетевые услуги (удаленный доступ, передача файлов, электронная почта).

1.3. Лабораторные занятия, их содержание и объем в часах.

ТЕМАТИЧЕСКИЙ ПЛАН ЛАБОРАТОРНЫХ ЗАНЯТИЙ

Наименование темы	**Кол-во часов**
1. Основы программирования на Java. Создание простейших апплетов	4
2. Рисование в окне, обработка событий мыши и клавиатуры	4
3. Создание графического интерфейса пользователя	6
4. Контейнеры компонентов GUI и менеджеры размещения	6

5. Многопотоковые приложения	8
6. Разработка приложений	8
Итого	**36**

1.4. Самостоятельная работа студентов

В качестве самостоятельной работы по дисциплине «История и методология информатики и вычислительной техники » студенты готовят рефераты по следующим темам:

- Домеханическая эра
- Механическая эра
- Принцип программного управления Бэббиджа
- Математический фундамент
- Электронно-механические устройства
- Ламповые ЭВМ
- Развитие элементной базы ЭВМ
- От мини до супер-ЭВМ
- Микропроцессоры
- Персональные компьютеры
- Эволюция программного обеспечения

1.5. Вопросы к экзамену

1. Понятия информации и общая характеристика процесса сбора, хранения, обработки, защиты и передачи информации.
2. Методологические основы информатики. Модель - алгоритм - программа.
3. Место компьютера в современном мире: наука, бизнес, искусство, системы связи, экономика, управление, война, досуг и т.д.
4. Формальные модели алгоритмов, проблемы вычислимости, сложность вычислений.
5. Базы данных (структуры данных, поиск ответов на запросы, логический вывод).
6. Инженерия математического обеспечения (языки программирования, технологии создания программных систем, инструментальные системы).
7. Искусственный интеллект (представление знаний, вывод на знаниях, обучение, экспертные системы).
8. Распознавание образов и обработка зрительных сцен.
9. Компьютерная лингвистика (модели языка, анализ и синтез текстов, машинный перевод).
10. Числовые и символьные вычисления (компьютерно-ориентированные методы вычислений, модели переработки информации в различных прикладных областях, работа с естественно-языковыми текстами).

11. Нейросистемы (теория формальных нейронных сетей, использование нейронных сетей для обучения, нейрокомпьютеры).
12. Меры информации, понятие «количество информации», пропускная способность канала.
13. Избыточность информации, суть эффективного кодирования.
14. Каким образом кодирование защищает сообщение от воздействия помех?
15. Принципы фон Неймана.
16. Принцип открытой архитектуры.
17. Что такое компьютерные вирусы?
18. Для чего нужны системные библиотеки?
19. Чем отличаются компиляторы от интерпретаторов?
20. Какова хронология развития языков программирования?
21. Что такое машинный язык?
22. Какие языки программирования называются языками высокого уровня?
23. В чем преимущества и языков программирования низкого уровня над языками высокого уровня?
24. Что такое исходный код?
25. Особенности машинно-зависимых языков программирования.
26. Чем отличаются языки символического кодирования?
27. Расскажите об автокодах.
28. В чем особенности проблемно-ориентированных языков?
29. В чем особенности процедурно-ориентированных или алгоритмических языков?
30. Каково назначение непроцедурных языков программирования?
31. Какие диалоговые языки программирования Вам известны?
32. Какие функции выполняют языки описания сценариев?
33. Что такое язык Ассемблера?
34. Как организовать проверку программы?
35. Что такое программирование сверху-вниз?
36. Парадигма программирования.
37. Дайте определение процедурного или императивного программирования?
38. В чем смысл функционального или аппликативного программирования?
39. Особенности логического или реляционного программирования.
40. Дайте понятие объектно-ориентированного программирования.
41. Понятие «Информация» и «Знание». Их генезис.
42. Машина Тьюринга, многоленточные автоматы.
43. Модель фон Неймана; поколение ЭВМ; перспективы развития ЗУ, БД, процессоров.

1.6. Виды контроля.

Для проверки эффективности преподавания дисциплины проводится контроль знаний студентов. При этом используются следующие виды контроля:

- *текущий контроль* за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения аудиторных занятий посредством устного опроса, проведения контрольных работ или осуществления лекции в форме диалога.

- *промежуточный контроль* осуществляется два раза в семестр в виде анализа итоговых отчетов на аттестационные вопросы.

- *итоговый контроль* в виде зачета осуществляется после успешного прохождения студентами текущего и промежуточного контроля и сдачи отчета по самостоятельной работе и экзамена устного или письменного при ответах экзаменуемого на два вопроса в билете и дополнительные вопросы по желанию экзаменатора.

1.7. Требования к знаниям студентов, предъявляемые на экзамене

Для получения экзамена посещать занятия, проявлять активность в аудитории, обязан выполнить все лабораторные работы, знать теоретический материал в объеме лекционного курса, защитить отчет по самостоятельной работе.

2. Учебно-методические материалы по дисциплине

2.1. Перечень обязательной (основной) литературы

1. Анализ алгоритмов. Активный обучающий подход: учеб.пособие/ Дж. Маконнелл; пер. с англ. С.А. Кулешов.-3-доп.изд..М:.; Техносфера, 2009. – 416 с.

2. Бройдо В.Л. Архитектура ЭВМ и систем: учеб. Рек.Мин.обр. РФ / Бройдо, В.Л., Ильина, О.П. -2-е изд.- СПб. [и др.]: Питер, 2009. - 720 с.

3. Таненбаум Э. Компьютерные сети.- 4 –е изд. – СПб.: ПИТЕР, 2006. – 992 с.

2.2. Перечень дополнительной литературы

1. Воройский Ф.С. Информатика: введение в современные информационные и телекоммуникационные технологии в терминах и фактах.Энциклопедический словарь-справочник. – М.: Физматлит, 2006. – 768 с.

2. Брукшир Дж.Г. Информатика и вычислительная техника.- 7-е изд. – СПб.: Питер, 2004. – 620 с.

3. С.В. Симонович и др. Информатика: Базовый курс. СПб.: Питер. – 2007. –640 с.

3. Необходимое техническое и программное обеспечение

Лекции проводятся в стандартной аудитории, оснащенной в соответствии с требованиями преподавания теоретических дисциплин.

Для проведения лабораторных работ необходим компьютерный класс на 10 посадочных рабочих мест пользователей. В классе должен быть установлен язык программирования Java.

4. Учебно-методическая (технологическая) карта дисциплины

Номер недели	Номер темы	изучаемые на лекцииВопросы,	Занятия		и методические пособияИспользуемые наглядные	Самостоятельная работа студентов		Форма контроля
			Практические	Лабораторные		*Содержание*	*Часы*	
1	2	3	4	5	6	7	8	9
1	1	1-5		1	2.1.	Выбор темы самостоятельной работы	5	злр
2	2							
3		7-9		2	2.2	Поиск литературы по самостоятельной работы	10	злр
4	3							
5					2.1.			злр
6	4	10-12						
7		15-17		3	2.1.	Работа с литературой и поиск информации в сети Интернет	24	злр
8								
9	5				2.1.			злр, сб.
10				4				
11		13-16			2.2	Написание реферата	6	злр
12	6			5				
13					2.2			злр, защ.
14								
15	7	18-20			2.2		4	злр

16 17 18					2.1	Защита реферата по самостоятельной работе		злр, защ.

2. ГРАФИК САМОСТОЯТЕЛЬНОЙ УЧЕБНОЙ РАБОТЫ СТУДЕНТОВ

Понятие «самостоятельная работа» имеет две стороны: во-первых, это единственный метод усвоения знаний, во-вторых, это одна из организационных форм обучения.

Самостоятельная работа студентов включает следующие виды работ:

- подготовку к лабораторным работам, зачету;
- работу с периодическими изданиями, с нормативно-правовой документацией;
- текущие консультации.

3. КОНСПЕКТ ЛЕКЦИЙ ПО ДИСЦИПЛИНЕ

Период до первых ЭВМ.

В истории механических вычислителей заметными датами оказались 1617 г., когда шотландец Д.Непер описал устройство для сложения и умножения, напоминающее счеты; 1642 г. – год изобретения французом Б.Паскалем суммирующей машины; 1694 г. – немец Г.Лейбниц создал машину, умеющую и складывать, и умножать; 1874 г. – год появлении арифмометра, сконструированного петербуржцем В.Однером. Следует отметить также работающую модель 6-ти разрядного механического вычислительного устройства, которое могло складывать и вычитать числа, созданную немцем Вильгельмом Шиккардом (Schickard) (1592-1635). Но наиболее заметный след в истории механических вычислителей оставил Ч.Бэббедж.

Чарльз Бэббедж (1791-1871) - английский изобретатель, автор первой в мире аналитической счетной машины, являющейся прообразом современных вычислительных машин с программным управлением.

Бэббедж окончил университет в Кэмбридже. В 1820 г. он увлекся разработкой разностной счетной машины для составления таблиц многочленов. Работа машины основана на том, что *n*-я разность многочлена *n*-й степени является постоянной. Поэтому, зная несколько начальных значений функции для равноотстоящих значений аргумента, можно рассчитать конечные разности вплоть до постоянной *n*-й и выполнить обратный ход - по разностям вычислить новое значение функции. Циклически повторяя расчет, можно получить таблицу функции с любым числом строк (значений аргумента). В машине Бэббеджа эти вычисления выполнялись автоматически с помощью совокупности вращающихся колес.

В 1834 г. Бэббедж разрабатывает основные принципы построения универсальной машины, названной им аналитической. Именно этот проект стал описанием первой в мире универсальной вычислительной машины.

Оба проекта Бэббеджу не удалось довести до завершения из-за трудностей финансового характера.

Электрорелейные компьютеры предшествовали появлению ЭВМ и создавались в первой половине 40-х годов прошлого века. Наиболее известны электрорелейные машины К.Цузе (Германия) и Г.Айкена (США).

В проекте вычислителя Z-3, созданного в 1941 г. в Германии Конрадом Цузе (1910-1995), использованы двоичное представление информации и преобразование десятичных кодов в двоичные, выполнялось 8 команд, в число которых входили 4 арифметических действия и извлечение квадратного корня. Операции выполнялись с плавающей запятой. Время сложения составляло 0,3 с, умножения - 4 с, емкость памяти (на релейных схемах) состояла из 64 22-разрядных чисел, 7 разрядов отводились для порядка и один разряд - для знака числа. Программа хранилась на перфоленте. Машина применялась главным образом для проверочных расчетов в области аэродинамики.

Одна из модификаций модели "Z-3" с фиксированным алгоритмом в течение двух лет функционировала в контуре системы автоматического управления технологическим процессом на линии сборки летающих снарядов. Для этого применялось устройство считывания данных с объекта и преобразования их в цифровую форму. Этим было положено начало использованию ЦВМ в качестве управляющих вычислительных машин.

Большую известность в силу субъективных причин получила машина Марк-1 (1944 г.), созданная Говардом.Айкеном (Aiken Howard) (1900-1973) на механических и электрорелейных элементах наподобие машины Ч.Бэббеджа. Сложение и вычитание в Марк-1 осуществлялись на 72 механических счетчиках по 24 цифровых колеса каждый.

Электромеханическая ЦВМ Марк-1

Первые зарубежные ЭВМ. Принято различать поколения ЭВМ: 1-е поколение - ламповые ЭВМ, 2-е поколение - полупроводниковые ЭВМ, 3-е поколение - ЭВМ с элементной базой на интегральных схемах, 4-е поколение - ЭВМ с элементной базой на БИС и СБИС.

Американский ученый Дж. фон Нейман – автор ряда основополагающих идей в области вычислительной техники. Именно с его именем связывают основные архитектурные принципы ЭВМ первых поколений.

Джон фон Нейман родился 28 декабря 1903 г. в Будапеште. После окончания высшей школы Нейман в течение двух лет изучал химию в Берлинском университете, а затем также в течение двух лет в Цюрихе. В 1927 г. стал приват-доцентом Берлинского университета. С 1929 г. фон Нейман живет в Принстоне (США). В 1933 г. он приглашен работать в математический отдел Института высших исследований.

Деятельность фон Неймана была чрезвычайно многообразной. Основная сфера его деятельности - математика, в которой ему принадлежит ряд крупных достижений. Но заметны результаты, полученные им и в физике. Широкую известность получил взрывной метод инициирования атомного взрыва, предложенный Фон Нейманом независимо от других. Фон Нейман был тесно связан с работами по использованию ядерной энергии, отдавал много времени, энергии и сил укреплению военной мощи своей новой родины. Последние годы его жизни были полностью посвящены работе в правительственных учреждениях. Он умер от саркомы 8 февраля 1957 г.

Безупречная логика была наиболее характерной чертой его мышления. Высказывание одного из его коллег: "Слушая фон Неймана, начинаешь понимать, как должен работать человеческий мозг". Другой отличительной чертой его ума была замечательная память.

В 1936 г. в Принстон приехал на два года заниматься математической логикой англичанин Алан Тьюринг (1912-1954). Здесь он опубликовал свою знаменитую работу об универсальных вычислительных машинах, после которой в учебники по языкам и алгоритмам вошел термин «машина Тьюринга», показывающая принципиальную возможность решения любых задач с помощью элементарных арифметических действий. Фон Нейман предложил Тьюрингу место ассистента для совместной работы. Но Тьюринг вернулся в Англию, где в годы войны стал искусным дешифровальщиком немецких сообщений.

Интерес фон Неймана к компьютерам непосредственно связан с его участием в Манхэттенском проекте по созданию атомной бомбы, который разрабатывался в ряде мест США, в том числе и в Лос-Аламосе.

.В 1953 г. фон Нейман присоединился к группе Д.Моучли и Д.Эккерта, разрабатывавших машину ЭНИАК. А через год им подготовлен отчет, в котором обобщены планы работы по созданию компьютера EDVAC с архитектурой, получившей название фоннеймановской (хотя идея хранения программы в памяти машины уже была использована Моучли и Эккертом и высказывалась Тьюрингом).

Основные принципы фоннеймановской архитектуры:

- естественный (последовательный) порядок выполнения команд;

- хранение в памяти как чисел, так и команд;

- команды содержат адреса операндов.

Первой электронной вычислительной машиной обычно называют ЭНИАК (Electronical Numerical Integrator and Calculator), разработка которой велась под руководством Д.Моучли (John Mauchly) (1907-1980) и Д.Эккерта (John Eckert) (1919-1995) и закончилась в 1946 г., хотя приоритет Моучли и Эккерта оспорен Д.Атанасовым. Машина ЭНИАК была установлена в Пенсильванском университете. Она состояла из 18000 электронных ламп и 1500 реле и потребляла около 150 кВт электроэнергии. Программное управление последовательностью выполнения операций осуществлялось как в счетно-аналитических машинах с помощью штекеров и наборных полей. Настроить ENIAC на какую-нибудь задачу означало вручную изменить подключение 6000 проводов. Все эти провода приходилось вновь переключать, когда нужно было решать другую задачу.

Однако приоритет создания первой ЭВМ был решением суда в 1973 г. отдан американскому ученому болгарского происхождения Джону Атанасову.

В конце 30-х годов Джон Атанасов (1903-1995), профессор колледжа штата Айова, после попыток создания аналоговых устройств для производства сложных вычислений начал работать над созданием цифрового компьютера с использованием двоичной системы счисления. Машина строилась на электромеханических и электронных компонентах. Атанасов изобрел, в частности, регенеративную память на конденсаторах. При помощи аспиранта К.Берри он построил опытный образец машины для решения дифференциальных уравнений. Машина получила название Эй-Би-Си. В 1941 г. молодой Д.Моучли побывал в Пенсильванском университете, где познакомился с машиной Атанасова и изучил документацию к ней. Атанасов готовил заявку на получение патента на созданную им с Берри машину. Но вскоре он был направлен на работу в одну из лабораторий военно-морских сил США и заявка так и не была подана. В 1946 г. рассекретили ЭНИАК. И уже Моучли и Эккерт подали ряд патентных заявок, связанных с ЭНИАКом.

Атанасов стал отстаивать свой приоритет лишь тогда, когда случайно из газет узнал по фотографии давнего визитера в его лабораторию. В 1973 г. коллегия Миннеаполисского окружного суда постановила, что Моучли использовал идеи, составившие основу поданных патентов и ставшие ему известными благодаря давнему визиту к Атанасову. Первым электронным компьютером суд признал не ЭНИАК, а Эй-Би-Си. Справедливости ради нужно заметить, что специализированное устройство Эй-Би-Си было экспериментальным компьютером, а ЭНИАК активно использовался до 1955 г.

В 1951 году Эккерт и Моучли разрабатывают UNIVAC I (Universal Automatic Computer), предназначенный для решения разнообразных задач бизнеса. Затем было создано несколько разных моделей UNIVAC, которые нашли применение в различных сферах деятельности. Таким образом, UNIVAC стал первым серийным компьютером. Можно сказать, что UNIVAC положил начало компьютерному буму.

Еще одной машиной-предшественником ENIAC, кроме машины Атанасова, является британский компьютер Colossus («Колосс»), введенный в эксплуатацию в 1943 г. Главный конструктор машины Томми Флауэрс (Tommy Flowers). Этот компьютер мало известен из-за сверхсекретности его применения.

Среди других значительных разработок британских инженеров выделяется созданный в 1963 г. под руководством Килбурна в компании, позднее получившей название ICL (International Computers Limited), компьютер ATLAS .

Компьютеры, разрабатывавшиеся в компании IBM**.** В развитии вычислительной техники в США главные роли играли такие компании, как IBM, Hewlett-Packard (HP), CDC, Intel и ряд других

Компания IBM - мировой лидер в области создания средств вычислительной техники - основана в 1896 г. под названием Tabulating Machine Company изобретателем бумажных перфокарт и табуляторов Г. Холлеритом (1860-1929). В 1914 г. генеральным менеджером компании стал Томас Дж. Уотсон-старший, с именем которого связаны основные достижения компании в 20-40-х годах. С 1924 г. компания носит имя International Business Machines (IBM).

В начале 40-х годов прошлого века в лабораториях IBM совместно с учеными Гарвардского университета (во главе с Г.Айкеном) была начата и закончена в 1944 г. разработка одной из первых электромеханических вычислительных машин "Марк-1".

В 1953 г. была выпущена IBM 701, построенная на электронно-вакуумных лампах, с быстродействием до 17 тыс. оп./с, с памятью на электронно-лучевых трубках, изобретенной Ф.Вильямсом (Frederic Williams) (1911-1977) и Е.Килберном в том же году [10]. Инициатором создания этой машины, названной Defence Calculator, что подчеркивало ее оборонное назначение, был Томас Дж. Уотсон-младший (президент IBM с 1952 г.). Именно ему принадлежит инициатива перехода IBM к производству компьютеров, хотя перспективы рынка были тогда весьма туманными: считалось, что спрос на них составит в стране всего несколько штук. Разработку IBM 701 выполнило подразделение прикладных исследований IBM, которым руководил Гутберт Херд. В совершенствовании IBM 701 и 702 участвовали такие ведущие специалисты, как Джин Амдал и Джон Бэкус, впоследствии активно занимавшиеся разработкой компьютера IBM 705.

В 1953 г. американский инженер и эксперт по менеджменту Джей Форрестер (Jay Forrester) изобрел запоминающее устройство на ферритовых сердечниках, которое стало использоваться в качестве оперативной памяти вместо потенциалоскопов.

В 1955 г. появилась ЭВМ IBM 705, главным конструктором которой был Джин Амдал. Он же разработал операционную систему для этой машины.

Появление параллелизма можно отнести к 1958 г., когда Д.Слотник предложил проект SOLOMON - первой машины типа SIMD.

В 1959 г. IBM выпускает ЭВМ второго поколения IBM 1401, а затем создает свой первый мэйнфрейм IBM 7090 с быстродействием 229 тыс. оп./с.

В 1964 г. IBM выпускает первые модели System/360 (иначе IBM-360) [11], назвав эту серию компьютерами третьего поколения, первые машины были на гибридных микросхемах. В разработке участвовали Д. Амдал, Г. Блау, Ф. П. Брукс-младший. Ряд System/360 был грандиозным проектом (стоимость 30 млрд. долл., было задействовано около 100 тыс. сотрудников IBM), но его эффективность имела неоднозначную оценку.

Джин Амдал, подобно С.Крею, является легендарной личностью в истории компьютерной индустрии. После успешной разработки System/360 в 1970 г. Амдал решил создать свою собственную фирму, в которой, начиная с 1974 г., он проектирует ряд мощных IBM-совместимых мэйнфреймов. Мэйнфреймы Amdahl 470v/6, Amdahl 5860, серверы Millennium и др. успешно конкурировали с популярными мэйнфреймами IBM. Первое применение матричных БИС относится к 1976 г., это сделал Д.Амдал в машине Amdahl 470v/6. Как правило, компьютеры Amdahl обладали более высокой производительностью, чем аналогичные машины IBM, при сравнимой, а то и более низкой цене.

Модель 50 System/360

С 1971 г. IBM предлагает модели семейства System/370 на монолитных интегральных схемах. Запуском в производство новых моделей семейства 370 руководил Т. В. Лерсон, сменивший в 1974 г. Т. Дж. Уотсона-младшего на посту президента IBM.

В 1980 г. в исследовательском центре им. Томаса Дж. Уотсона был создан IBM 801 Minicomputer - первый компьютер, оснащенный прототипом RISC-процессора. Руководитель проекта IBM 801 Джон Кок был одним из авторов концепции RISC-архитектуры. В IBM 801 была реализована суперскалярная архитектура на серийных КМОП-микросхемах, позволяющая выполнять параллельно несколько команд на независимых функциональных устройствах.

В 1981 г. корпорация IBM выпустила на рынок свой первый персональный компьютер IBM PC. Разработку IBM PC выполнила группа из 12 инженеров IBM под руководством Вильяма Си Лоува. При этом были использованы разработки других фирм: микропроцессор i8088 корпорации Intel, операционная система DOS корпорации Microsoft.

В 1988 г. появилось семейство компьютеров IBM AS/400, применяемое преимущественно в качестве серверов баз данных, серверов банковских транзакций и т. п. Конфигурации серверов на базе AS/400 могут быть многопроцессорными, причем AS/400 относится к числу наиболее производительных SMP-систем.

В 1990 г. были выпущены мэйнфреймы семейства 390, которые, как и все предыдущие модели семейств System/360 и System/370, поддерживали совместимость приложений "снизу-вверх".

Развитие суперскалярной архитектуры получило развитие в 1990 г. в компьютерах RISC System/6000 (IBM RS/6000). Это архитектура POWER (Performance Optimization with Enhanced RISC). Тогда же была представлена версия операционной системы Unix, названная AIX Version 3.

Во второй половине 90-х годов началось вытеснение ЭСЛ-схемотехники КМОП элементной базой. Впервые КМОП-схемы в мэйнфреймах стал применять Д.Амдал.

В 2000 г. в IBM разработано уже шестое поколение мэйнфреймов серии 390 - S/390 G6.

Персональные компьютеры.

Предшественниками персональных компьютеров были электронные калькуляторы. Один из них Altair (1974 г.), разработанный Эдвардом Робертсом (Roberts) в компании MITS, иногда называют первым персональным компьютером, хотя это преувеличение, поскольку ввод информации в Altair осуществлялся в двоичном виде с помощью тумблеров, а при выключении компьютера вся информация терялась.

Идея персонального компьютера (точнее, ноутбука) зародилась в Исследовательском центре фирмы Xerox. Ее автором является Алан Кей, работавший в этом Центре с 1972 г. Здесь А.Кей стал заниматься проблемами безбумажных технологий и интерактивных режимов работы с компьютером. В 1977 г. им опубликовано описание портативного интерактивного устройства с плоскопанельным сенсорным экраном, беспроводной системой коммуникации и мультимедийными возможностями. Кей назвал это устройство Dynabook. К сожалению, проект Dynabook не был завершен. В дальнейшем Кей работал в компании Apple, основанной С.Джобсом и в которой были созданы персональные компьютеры Macintosh.

В 1976 г. Стивом Джобсом и Стефаном Возняком была основана компания Apple Computer. Молодые люди, не имевшие законченного высшего образования, решили собрать первый персональный компьютер (ПК) и поскольку руководство фирм, в которых они работали, изобретением не заинтересовалось, решили основать собственную компанию. Преодолев финансовые трудности, вдвоем в помещении отцовского гаража С.Джобс с коллегой собрали первые ПК Apple I. Видные специалисты (например, Р.Нойс, Б.Хьюллет) не сумели правильно оценить перспективы ПК. Несмотря на это, Apple I имел большой успех. Еще больший успех пришелся на долю следующего ПК Apple II (1977 г.) с цветным экраном. Джобс стал весьма популярен в США, ошеломляющий успех, фотогеничность, мастерство оратора привлекали к молодому руководителю Apple Computer все большее внимание со стороны общественности. Но в 1985 г. Джобс (из-за разногласий с руководителем компании Скэлли, которого пригласил в Apple сам Джобс) покинул Apple и основал компанию NextStep. Однако ни Apple, ни NextStep не смогли соперничать с тандемом IBM – Microsoft и компьютеры Apple Макинтоши отошли в тень. Позднее для спасения Apple Computer С.Джобс вернулся в компанию и ситуация несколько стабилизировалась.

Однако непосредственным разработчиком Apple I и Apple II был не С.Джобс, а С. Возняк. Именно он спроектировал эти машины, а С.Джобс больше выполнял функции менеджера. В феврале 1981 года С.Возняк попал в авиакатастрофу. Чудом уцелев, он серьезно повредил лицо. Выйдя из больницы, он сразу же уволился из Apple, затем женился и поступил в университет Беркли, чтобы продолжить незаконченное образование. Далее он пытался создавать собственные компании, но малоуспешно. В последнее время занимается преподаванием и благотворительностью.

На звание первого персонального компьютера в истории претендуют такие машины как MITS Altair, PET Commodore, TRS-80, но только Apple II раньше остальных стал поставляться в пластиковом корпусе вместе с цветным дисплеем и алфавитно-цифровой клавиатурой.

Серийный Apple I

В 1980 г. компания Hewlett-Packard: выпускает свой первый персональный компьютер HP-85.

В 1981 г. появляется первый персональный компьютер IBM PC компании IBM.

Первым карманным персональным компьютером стал Pilot, созданный Джефом Хокинсом (Jeff Hawkins) в 90-е годы.

Архитектуры ЭВМ. В 1966 г. М.Флинн (Michael Flynn) предложил классифицировать вычислительные системы по соотношению потоков команд и данных. Этот подход используется до наших дней.

Основоположником архитектуры ЭВМ, называемой компьютер с полным набором команд (Complete Instruction Set Computer - CISC), считают компанию IBM с ее базовой архитектурой IBM-360, ядро которой используется с 1964 года. К классическим CISC-архитектурам можно отнести архитектуру VAX. Микропроцессоры компании Intel (архитектурный ряд x8б и Pentium) достаточно близки к данной архитектуре.

Для CISC-процессоров характерно:

• сравнительно небольшое число регистров общего назначения;

• большое количество машинных команд, некоторые из которых функционально аналогичны операторам высокоуровневых языков программирования и выполняются за несколько тактов;

• большое количество методов адресации;

• большое количество поддерживаемых форматов команд различной разрядности;

• преобладание двухадресного формата команд.

Архитектура, называемая компьютер с сокращенным набором команд (Reduced Instruction Set Computer - RISC), появилась благодаря тому, что еще в середине 70-х годов некоторые разработчики компьютерных архитектур заметили, что даже у компьютеров сложной архитектуры большая часть времени уходит на выполнение простых команд. Это наблюдение легло в основу работ по созданию IBM 801 - первой RISC-машины, разработка которой была завершена в 1979 г.

Само понятие RISC было введено Дэвидом Паттерсоном (David Patterson), преподавателем университета Беркли, в 1980 году.

Основными чертами концепции RISC-архитектуры являются:

• одинаковая длина команд;

• единый формат команд, или, по крайней мере, использование не более двух-трех форматов;

• операндами всех арифметических и логических команд могут быть только регистры;

• команды выполняют только простые действия;

• выполнение любой команды производится не дольше, чем за один такт;

• большой регистровый файл;

• только простая адресация.

В системах программирования для RISC-архитектуры практически всегда присутствуют оптимизирующие компиляторы.

Архитектура с длинным командным словом (Very Long Instruction Word - VLIW) - это статическая суперскалярная архитектура. Несколько простых команд упаковывается компилятором в длинное слово. Слово соответствует набору функциональных устройств. Распараллеливание кода производится на этапе компиляции, и в машинном коде уже присутствует явный параллелизм.

Примером реализации VLIW-архитектуры является суперкомпьютер Эльбрус-3. Собственно длинное командное слово занимает в нем 256 бит в упакованном виде и до 500 бит в распакованном представлении. Микрораспараллеливание на уровне операций обеспечивается как для скалярных, так и для векторных вычислений. В каждой команде в максимальном варианте может запускаться одновременно до 7 арифметико-логических операций. В качестве аргументов они могут использовать результаты ранее выполненных операций, поступающие непосредственно с выхода арифметико-логических устройств, либо из быстрой регистровой памяти (буферы стека емкостью 1024 слова или буферы считанных элементов массивов емкостью 512 слов). Предусмотрено одновременное параллельное обращение по 8 каналам в локальную и (или) глобальную память.

Развитием данной архитектуры являются вычисления с явным параллелизмом команд (Explicitly Parallel Instruction Computing - EPIC). Концепция EPIC разработана совместно компаниями Intel и Hewlett-Packard. Она обладает достоинствами VLIW, но лишена ее недостатков (например, использует специальные механизмы для исключения неэффективности кодов традиционных VLIW-архитектур, требовавших применения излишних команд для заполнения пустых машинных тактов). Данная архитектура имеет следующие основные особенности:

• масштабируемость архитектуры до большого количества функциональных устройств;

• явный параллелизм в машинном коде;

• предикатное выполнение команд, исключающее переходы. Команды из разных частей условного ветвления снабжаются предикатными полями и запускаются параллельно.

Зарубежные суперкомпьютеры. Первый транзисторный суперкомпьютер CDC 1604 создан Сеймуром Креем (Seymour Cray) в 1958 г. в компании Control Data Corporation (CDC). В этой компании С.Крей принимал участие также в разработке машин серии Cyber.

В дальнейшем развитие суперкомпьютеров связано с реализацией идей параллелизма. О параллельной обработке данных в свое время высказывался Ч.Беббидж. Первые проекты многопроцессорных ЭВМ были описаны в статьях Д.Холланда (1959 г.), Д.Слотника (1962 г.) и др.

Наибольшую известность на стыке 60 и 70-х годов получил ILLIAC IV (закончен к 1972 г.) - система с 16 процессорами [12], разработанная в корпорации Burroughs. Это первый компьютер, в котором использовалась быстрая память на микросхемах. Быстродействие такого компьютера достигало 150-200 Мфлопс. По производительности ILLIAC IV существенно превосходил БЭСМ-6, но столь же существенно уступал советской машине по соотношению цена/производительность. Демонтирован ILLIAC IV был только в 1983 году.

В 1976 году началось производство первых в мире векторно-конвейерных суперкомпьютеров Cray-1. Эта машина была создана небольшим коллективом под руководством Сеймура Крея, который после того, как CDC решила прекратить работу над суперкомпьютерами, основал (1972 г.) свою компанию Cray Research для создания таких машин.

С.Крей (1925-1996) - американский специалист, признаваемый в мире, как первый разработчик суперкомпьютеров.

С.Крей - участник второй мировой войны. После демобилизации он получает степени бакалавра по электрике и магистра по математике. Работая в компании ERA, позднее превратившейся в Remington Rend, Крей участвует в разработках, положивших начало компьютерам ILLIAC. Но сверх основных обязанностей, выполняемых им в компании, он начинает конструировать свой компьютер. При этом С.Крей принимает ряд удачных решений. В частности, он использует принципы RISC-технологии еще до того, как эта технология была признана.

В результате сокращения финансирования Remington Rend вскоре переориентировалась на коммерческие компьютеры. Чтобы продолжить работу над линией ЭВМ для научных целей, С.Крей решил присоединиться к Б.Норису, создававшему в то время компанию CDC (Control Data Corporation).

Под руководством С.Крея в CDC ведется разработка и сборка больших машин, предназначенных для научно-исследовательских задач. Первый компьютер Крея - CDC 1604 был собран на транзисторной базе. Тем самым Крей доказал, что транзисторы, могут с успехом использоваться для построения вычислительных машин. Следующими двумя компьютерами Крея были большие ЭВМ CDC 6600 и CDC 7600, широко использовавшимися для решения научных задач.

В 1972 г. С.Крей решил основать собственную фирму Cray Research Inc. и приступить к проектированию ЭВМ Cray-1 [13], построенной на интегральных схемах ЭСЛ типа. Объем памяти этой машины 8 Мбайт, поделенных на 16 блоков емкостью 64К 48-разрядных слов каждый, с суммарным временем доступа 12,5 нс. Имелась и внешняя память на магнитных дисках емкостью около 450 Мбайт, расширявшаяся до 8 Гбайт. Для машины был создан оптимизирующий транслятор с Фортрана, макроассемблер и

специальная многозадачная ОС. На некоторых классах задач производительность доходила до 160 Мфлопс. Следует отметить, что Cray-1 был заметно дешевле, чем военные системы типа ILLIAC IV. Первый Cray-1 имел оригинальную конструкцию, в которой минимизировались длины проводников. Cray-1 стоил 8,8 млн долларов и был установлен в Национальной лаборатории в Лос-Аламосе.

. Cray-1

Далее последовали разработки Cray-2 и Cray-3. Суперкомпьютеры Cray-2 (1985 г. быстродействие 2 млрд. оп./с) и Cray-3 (1989 г., 5 млрд. оп./с) были самыми производительными суперкомпьютерами мира в то время.

Тактовая частота Cray-3 опять рекордная - 500 МГц, что достигнуто благодаря использованию арсенид-галлиевых микросхем. Но к этому моменту столь мощные компьютеры еще не были востребованы и Cray-3, как и Cray 4 с частотой в 1 ГГц, так и не были проданы.

С.Крей погиб в автомобильной катастрофе в 1996 г.

К концу 80-х годов холодная война закончилась и финансирование военных проектов, неразрывно связанных с суперкомпьютерами, в США временно сократилось. Лидерство на мировом рынке сразу же захватили энергичные японские фирмы — Fujitsu, Hitachi и NEC. Предложенная ими коммерческая концепция распределенных вычислений в среде из множества дешевых микропроцессоров (в настоящее время признанная в Японии стратегической) быстро себя оправдала.

Летом 1995 г. два токийских университета продемонстрировали специализированный (предназначенный для моделирования задач астрофизики) суперкомпьютер GRAPE-4, собранный из 1692 микропроцессоров и обошедшийся всего в 2 млн. долл. Он первым в мире преодолел порог в 1 трлн. оп./с с результатом 1,08 Тфлопс. Через 15 месяцев компания Cray Research сообщила, что модель Cray T3E-900, насчитывавшая 2048 процессоров, побила рекорд японцев и достигла 1,8 Тфлопс. К тому времени результат NEC SX-4 составлял 1 Тфлопс, Hitachi SR2201 — 0,6 Тфлопс, а Fujitsu Siemens VPP700 — 0,5 Тфлопс.

В 1997 г. появились сообщения о проекте моделирования ядерного взрыва (ASCI) в Лос-Аламосской лаборатории, финансируемом министерством энергетики США. Созданный в соответствии с этим проектом комплекс ASCI Red на 9632 процессорах Pentium Pro компании Intel показал производительность сначала 1,8 Тфлопс, а затем 3,2 Тфлопс.

В 2002 г. в рамках ASCI временами удавалось добиться скорости обработки информации 10,2 Тфлопс. Был предложен проект поиска внеземных цивилизаций, объединяющий сотни тысяч пользователей ПК, предоставляющих ресурсы своих компьютеров для

системы распределенных вычислений grid, в котором достигнута уникальная пиковая производительность 92 Тфлопс.

Мировыми лидером по производительности среди суперкомпьютеров стали в 2002 г. суперкомпьютер ASCI White компании IBM с 8192 процессорами и с производительностью 7,3 Tfops, в ноябре 2003 г. - компьютер Earth-Simulator японской компании NEC с производительностью в 35,9 Tflops, установленный в Японии в 2002 г. и включающий 5120 процессоров, а в ноябре 2004 г. - компьютер IBM BlueGene/L с 70,7 Tflops .

ASCI White

В 2004 г. порог в 10 Tflops преодолел китайский суперкомпьютер "Шугуан-4000А", установленный в Шанхае.

Отечественные ЭВМ, созданные под руководством С.А.Лебедева в ИТМиВТ.

Основные универсальные ЭВМ первого и второго поколений разрабатывались в СССР по оригинальным проектам отечественных специалистов [2, 14]. Основные работы велись в ИТМиВТ, Киевском институте кибернетики, ИНЭУМ, СКБ-245.

Институт точной механики и вычислительной техники (ИТМиВТ) создан в 1948 г. Его директорами назначались видные ученые: 1948 г. Н.Г.Бруевич, 1950 г. М.А.Лаврентьев, 1953 г. С.А.Лебедев, 1974 г. В.С.Бурцев, 1986 г. Г.Г.Рябов.

В 1950 г. М.А.Лаврентьев пригласил С.А.Лебедева возглавить (по совместительству с работой Лебедева в Киеве) основную лабораторию института по разработке вычислительных машин. Именно в этой лаборатории были созданы проекты таких ЭВМ, как БЭСМ, БЭСМ-2, М-20, БЭСМ-6. В ней начали свой творческий путь видные ученые и разработчики вычислительных машин В.С.Бурцев, В.А.Мельников, Г.Г.Рябов, Б.А.Бабаян и др. После кончины С.А.Лебедева работы по созданию отечественных ЭВМ серии Эльбрус продолжали его ученики. Однако принятые на правительственном уровне решения о главном направлении развития вычислительной техники в стране на базе ЕС ЭВМ (по аналогии с серией американских машин IBM-360) сократило возможности продолжения оригинальных отечественных линий БЭСМ-6 и Эльбрус.

Сергей Алексеевич Лебедев (1902 - 1974) - основоположник отечественной вычислительной техники.

Родился в Нижнем Новгороде. Окончил МВТУ им. Н.Э.Баумана. Работал в МВТУ и во Всесоюзном электротехническом институте.

В 1946 г. С.А.Лебедев был приглашен на работу в Киевский институт электротехники и теплоэнергетики, где под его руководством в период 1948-1951 г.г. создавалась первая отечественная вычислительная машина МЭСМ. МЭСМ - малая электронная счетная машина первого поколения (1951 г.) Быстродействие 100 операций в с, представление чисел - с фиксированной запятой 16-ю двоичными разрядами, система команд - трехадресная. Имеются устройства арифметическое, управляющее, ввода/вывода, запоминающее на триггерах (емкость 31 число и 63 команды) и на магнитном барабане. Ввод с перфокарт или с штекерного устройства. Следовательно, архитектура и принципы построения МЭСМ были аналогичными тем, которые ранее уже использовались в ЭНИАКе, но которые, по-видимому, Лебедеву не были известны (в открытой печати сведения о фон-неймановской архитектуре появились позже). Машина включала 6000 электронных ламп и занимала отдельное крыло здания площадью 60 $м^2$. Потребляемая мощность 25 кВт.

Коллектив разработчиков насчитывал всего 12 специалистов и 15 техников. МЭСМ задумывалась лишь как макет будущей ЭВМ, но практически стала реально используемой машиной, поэтому первая буква М в аббревиатуре МЭСМ вместо "модель" электронной счетной машины стала интерпретироваться как «малая».

Параллельно с работой в Киеве С.А.Лебедев руководит разработкой большой электронной счетной машины БЭСМ в ИТМиВТ. С 1953 г. С.А.Лебедев возглавляет этот институт. Первой созданной здесь ЭВМ стала машина БЭСМ - большая электронная счетная машина первого поколения, разрабатывавшаяся в течение 1950-1953 гг. Производительность - 8-10 тыс. операций в с. Представление чисел - с плавающей запятой, 39 двоичных разрядов. Первая модель БЭСМ имела сниженное быстродействие около 2000 операций в с из-за отсутствия потенциалоскопов для спроектированной памяти, которые были заменены ртутными линиями задержки. В 1956 г. начались поставки потенциалоскопов, но уже в 1958 г. от них отказались, так как были разработаны магнитные накопители на ферритовых сердечниках. Новая модель получила наименование БЭСМ-2. Было создано 7 экземпляров БЭСМ-2 на Казанском заводе счетно-аналитических машин.

БЭСМ-4 - вариант БЭСМ на полупроводниковой элементной базе (главный конструктор О.П.Васильев, научный руководитель С.А.Лебедев). Быстродействие - 20 тыс операций/с, емкость оперативной памяти - 16384 48-разрядных слова. К 1962-1963 гг. относится создание прототипа, к 1964 г. - начало серийного выпуска.

М-20 (главный конструктор С.А.Лебедев) - одна из лучших машин первого поколения (1958 г.) Быстродействие - 20 тыс операций/с, разрядность 45 бит, внешняя память на магнитных барабанах и лентах. В этой машине применена первая операционная система ИС-2.

М-40 - компьютер (1960 г), считающийся первым Эльбрусом (на вакуумных лампах). быстродействие 40 тыс. оп/с. Главный конструктор С.А.Лебедев, его заместитель В.С.Бурцев. В 1961 г. зенитная ракета, управляемая компьютером М-40, на испытаниях успешно сбивает межконтинентальную баллистическую ракету, способную нести ядерное оружие.

Вершиной научных и инженерных достижений С.А.Лебедева стала БЭСМ-6, первый образец машины был создан в 1967 г. В ней реализованы такие новые принципы и решения, как параллельная обработка нескольких команд, сверхбыстрая регистровая память, расслоение и динамическое распределение оперативной памяти, многопрограммный режим работы, развитая система прерываний. БЭСМ-6 - суперЭВМ

второго поколения. Быстродействие - 1 млн операций/с, емкость оперативной памяти - 64-128К 50-разрядных слов. В аппаратуре БЭСМ-6 использовано около 60000 транзисторов и 180000 полупроводниковых диодов. Эта ЭВМ стала основной вычислительной системой для многих предприятий в оборонных отраслях промышленности и оставалась таковой в течение более полутора десятков лет. Всего в базовом варианте было выпущено около 350 компьютеров БЭСМ-6. В 1975 г. управление полетом по программе «Союз-Аполлон» обеспечивал вычислительный комплекс на основе БЭСМ-6.

В создании программного обеспечения для машин М-40, М-50 и БЭСМ-6 активное участие принимал Лев Николаевич Королев. Под его руководством был разработан комплекс управляющих программ для системы противоракетной обороны страны на базе ЭВМ М-40 и М-50, а в 1967 г - первая операционная система для БЭСМ-6, названная позднее "Диспетчер-68".

Необходимо также отметить малоизвестный (из-за соображений секретности) компьютер 5Э92б на дискретных транзисторах, созданный С.А.Лебедевым и В.С.Бурцевым в 1964 г. Его быстродействие 0,5 млн оп/с, емкость оперативной памяти 32 тыс. 48-разрядных слов. Использовался в первой советской противоракетной системе обороны Москвы.

БЭСМ-6

ЭВМ, созданные под руководством И.С.Брука и его учеников. Исаак Семенович Брук (1902-1974) - один из пионеров отечественной вычислительной техники.

И.С.Брук закончил МВТУ им. Н.Э.Баумана в 1925 г. (в одной группе с ним учился С.А.Лебедев). По окончании учебы работал во Всесоюзном электротехническом институте, на заводе в Харькове, с 1935 г. - в Энергетическом институте АН СССР (в 1958 г. переименован в ИНЭУМ - Институт электронных управляющих машин). Занимался разработкой механических и электронных аналоговых интеграторов. В 1948 г. вместе с Б.И.Рамеевым разработал проект цифровой ЭВМ, который так и не был реализован. К созданию электронных цифровых вычислительных машин И.С.Брук вернулся в 1950 г. после принятия на работу талантливых выпускников МЭИ, среди которых были будущие крупные ученые и разработчики ЭВМ Н.Я.Матюхин и М.А.Карцев. Первой ЭВМ, созданной под руководством И.С.Брука в единственном экземпляре, стала машина М-1 (главный конструктор Н.Я.Матюхин). ЭВМ М-1 сдана в эксплуатацию в начале 1952 г. и стала второй ЭВМ после МЭСМ в стране и первой в Москве. На ней решались важные научные и инженерные задачи. Время сложения в этой машине 20 мс, умножения 2 с. Емкость оперативной памяти - 512 25-разрядных слов. В машине насчитывается 730 электронных ламп.

Николай Яковлевич Матюхин (1927-1984) - один из видных создателей вычислительной техники в СССР, один из первых разработчиков САПР вычислительных систем и устройств.

Н.Я.Матюхин в 1950 г. окончил МЭИ и был направлен на работу в Энергетический институт АН СССР в лабораторию И.С.Брука, где молодой специалист сразу же стал фактическим главным конструктором ЭВМ М-1, а после ее пуска в эксплуатацию переключился на разработку новой машины М-3.

В 1957 г. Н.Я.Матюхин перешел в НИИ автоматической аппаратуры, где в качестве главного конструктора участвовал в разработке ряда специализированных вычислительных комплексов для управления в системах ПВО. Это ЭВМ "Тетива" (1962 г.), 5Э63 (1965 г.), 5Э76 (1973 г.) и вычислительный комплекс 65с180 (1976 г.) и др. Некоторые из этих комплексов выпускались вплоть до 1992 г., например, машин 5Э63-1 было выпущено 330 шт.

Заслугой Н.Я.Матюхина является создание первой в СССР системы автоматизированного проектирования средств вычислительной техники АСП-1 (1968 г.). В частности, в этой системе для логического моделирования цифровых устройств был предложен язык МОДИС.

На базе лаборатории И.С.Брука в 1958 г. создан Институт электронных управляющих машин, Брук стал его первым директором**.** После М-1 в лаборатории И.С.Брука создаются ЭВМ М-2 и М-3.

М-2 - универсальная вычислительная машина, созданная под руководством И.С.Брука и М.А.Карцева. Быстродействие - 2 тыс операций/с.

Михаил Александрович Карцев (1923-1983) - один из создателей отечественной вычислительной техники в XX веке.

М.А.Карцев - участник Великой отечественной войны. После войны окончил МЭИ. Работал в лаборатории И.С.Брука над созданием машины М-1. Главный конструктор ЭВМ М-2 и М-4 (1962 г.). Машина М-4 имела ряд модификаций.

В 1967 г. для реализации идей М.А.Карцева по созданию многомашинных вычислительных систем был создан НИИ вычислительных комплексов, который возглавил Михаил Александрович. В 1973 г. был сдан в эксплуатацию первый из нескольких десятков комплекс М-10. Это была машина третьего поколения с быстродействием 5 млн операций/с, с емкостью оперативной памяти 0,5 Мбайт. Комплекс малоизвестен, поскольку на его основе была создана система слежения за объектами в космосе с целью предупреждения о возможном ракетном нападении.

Следующий комплекс М-13 дорабатывался уже после смерти М.А.Карцева.

Машина М-3 - универсальная вычислительная машина, созданная в 1956 г. под руководством И.С.Брука и Н.Я Матюхина. Ее быстродействие - 1,5 тыс операций/с (с накопителем на ферритовых сердечниках). Документация на М-3 была передана в Ереван и Минск, где на основе М-3 спроектированы ЭВМ "Арагац", "Раздан" и Минск-1.

В.М.Глушков и ЭВМ, созданные под его руководством. Виктор Михайлович Глушков (1923-1982) - выдающийся ученый в области кибернетики.

После четырех курсов Новочеркасского индустриального института В.М.Глушков перешел на пятый курс Ростовского (на Дону) университета. По окончании университета (1948 г.) молодой специалист-математик был направлен на Урал. Работал ассистентом в Свердловском лесотехническом институте. В 1950 г. защитил кандидатскую диссертацию.

В 1956 г. по приглашению академика Б.В.Гнеденко переехал в Киев, став заведующим лабораторией вычислительной техники в институте математики АН УССР, т.е. возглавил ту лабораторию, которой до этого руководил уехавший в Москву С.А.Лебедев.

В Киеве Виктор Михайлович занимается разработкой теории проектирования ЭВМ. Начиная с 1958 г., ведутся разработки управляющей ЭВМ "Днепр" (главный конструктор Б.Н.Малиновский, научный руководитель В.М.Глушков), а с 1961 г. началось внедрение этих машин на заводах страны. Эти машины появились одновременно с управляющими машинами в США и выпускались целое десятилетие (обычно срок морального старения ЭВМ составляет пять-шесть лет).

В 1962 г. по инициативе В.М.Глушкова создается институт кибернетики АН УССР, а в 1963 г. - СКБ вычислительных машин. Создаваемые в этих организациях проекты ЭВМ воплощались на Киевском и Северодонецком заводах управляющих машин.

После "Днепра" главное направление работ коллектива под руководством Глушкова - создание интеллектуальных ЭВМ. Началось с машин, упрощающих инженерные расчеты. Это миниатюрные (по тем временам) Проминь (1962 г.) и Мир-1 (1965 г.). Вслед за ними появились более совершенные машина инженерных расчетов Мир-2 и машина Мир-3 с входным языком Аналитик, близким к обычному математическому языку. «Миры» успешно выполняли аналитические преобразования. Разработками Института кибернетики заинтересовались в США. Единственный случай покупки американцами советской ЭВМ относится именно к машине Мир-1.

Важным направлением работ Института кибернетики была автоматизация управления предприятиями. Конечной целью этих работ было создание ОГАС - общегосударственной автоматизированной системы управления экономикой.

Сложность управления экономикой растет сверхлинейно, и Глушков предвидел, что без АСУ на государственном уровне экономику страны в ближайшие годы ждет кризис. Еще в 1964 г. им разработаны основные предложения по созданию ОГАС. Частью ОГАС было развертывание единой сети вычислительных центров (в США проект подобной сети ARPANET появился позже). Требовались значительные усилия и средства для реализации грандиозной задачи. Однако в должной мере эти предложения не были реализованы по ряду причин. Во-первых, руководство страны не вполне понимало, что может дать широкая компьютеризация экономики и каким образом ее надо осуществить. А мнения ученых по этим вопросам не были однозначными. Прежде всего против выступали многие представители экономической науки, также не понимавшие сути компьютеризации. В отличие от них наши стратегические противники в США поняли, что с помощью современных математических методов и информационных технологий плановая социалистическая экономика может иметь преимущества перед капиталистической, что приведет к поражению США в "холодной" войне. За рубежом были осуществлены контрмеры: подготовлены и опубликованы статьи, дискредитирующие как идеи ОГАС, так и самого Глушкова. Несмотря на это, А.Н.Косыгин поддерживал Глушкова, однако после смерти Председателя Совмина СССР должной поддержки в Политбюро КПСС ОГАС не нашла.

Машины, созданные в московском СКБ-245 и в других организациях СССР.

Специальное конструкторское бюро № 245 (СКБ-245) создано в 1948 г., его основной задачей являлась разработка средств вычислительной техники в Советском Союзе. Начальником СКБ-245 был назначен М. А. Лесечко. Позднее М. А. Лесечко стал министром приборостроения и средств связи.

Одна из первых (наравне с БЭСМ) отечественных ЭВМ «Стрела» разрабатывалась в СКБ-245 министерства машиностроения и приборострооения СССР в 1950-1953 г.г. под руководством Ю.Я.Базилевского и Б.И.Рамеева. Быстродействие - 2000 операций/с, оперативная память 2048 43-разрядных слов. Машина трехадресная.

В группу разработчиков «Стрелы» входили Б.В.Анисимов, Д.А.Жучков, Н.В.Трубников, имена которых связаны с подготовкой инженерных кадров в МВТУ им. Н.Э.Баумана, Так, Б.В.Анисимов в 1952 г. основал и до конца жизни (1976 г.) руководил кафедрой «Математические машины».

Юрий Яковлевич Базилевский (1912-1983) был главным конструктором ЭВМ "Стрела", семь машин "Стрела" было изготовлено на Московском заводе САМ. В дальнейшем Ю.А.Базилевский руководил разработкой специализированных вычислительных комплексов М-111 и 5Э61 для оборонных систем, будучи главным инженером СКБ-245. В 1970–80-х годы работал в Минприборе заместителем министра

Башир Искандарович Рамеев (1918-1994) - талантливый конструктор электронных вычислительных машин, главный конструктор семейства (ряда) ЭВМ "Урал".

Б.И.Рамеев работал в ЦНИИ связи, служил связистом в армии. В 1947 г. узнав о разработке американцами машины ЭНИАК, по совету А.И.Берга перешел на работу в лабораторию И.С.Брука, вместе с которым в 1948 г. впервые в СССР разработал проект цифровой электронной вычислительной машины (проект не был реализован). В 1949 г. направлен для разработки ЭВМ начальником отдела в СКБ-245, где был одним из ведущих разработчиком ЭВМ «Стрела». «Стрела» была сдана комиссии в 1953 г. и отмечена Сталинской премией. С 1955 г. Б.И.Рамеев - главный конструктор машин "Урал" в Пензенском НИИ математических машин.

В 1962 г. Б.И.Рамеев без защиты диссертации стал доктором технических наук.

Машина Урал-1 - первая из серии ЭВМ «Урал», созданная в 1957 г. под руководством Б.И.Рамеева в СКБ-245. Эта малая машина отличалась дешевизной и потому получила сравнительно широкое распространение в конце 50-х годов. Быстродействие - 100 операций/с, оперативная память (1024 слова) - на магнитном барабане.

Вслед за Уралом-1 последовали Урал-2 с быстродействием 5000 операций/с с оперативной памятью на ферритовых сердечниках (1959 г.), Урал-11, Урал-14, Урал-16 - серия (ряд) аппаратно и программно совместимых ЭВМ второго поколения разной производительности. Эти машины создавались под руководством Б.И.Рамеева в 1962-64 гг. уже в Пензенском НИИ математических машин. Эта серия предвосхитила решения IBM-360, принятые в дальнейшем для разработки ЕС ЭВМ в странах СЭВ.

В 1967 г. Рамеев переходит работать начальником отдела в Научно-исследовательский институт электронных математических машин (НИЭМ), созданный в 1958 г. на базе СКБ-245.

За период 1958-1968 гг. в НИЭМ был разработан ряд ЭВМ как универсальных, так и специализированных для министерства обороны СССР. Заметной вехой в деятельности НИЭМ была разработка в период 1967-69 годов полупроводниковой ЭВМ М-222 коллективом под руководством Вениамина Степановича Антонова вместе с СКБ Казанского завода электронных вычислительных машин.

В.С.Антонов (1925-2004) - участник Великой отечественной войны, по окончании которой защитил диплом инженера в МАТИ. С 1950 г. работал в СКБ-245. После образования НИЦЭВТ В.С.Антонов активно участвовал в разработке ЕС ЭВМ. В 1973 г. под руководством главного конструктора В. С. Антонова была создана ЭВМ ЕС-1050, в 1977 г - ЕС-1060, при его участии отрабатывались ЭВМ ЕС-1066, двухмашинный комплекс ЕС-1068, операционные системы ОС-6 и ОС-7.

Одним из главных конструкторов ЭВМ в НИЭМ был директор института Сергей Аркадьевич Крутовских (1928-1981). В 1964 г. в НИЭМ впервые в СССР были развернуты работы по проектированию и производству бортовых ЭВМ, получивших название "Аргон". Первые образцы ЭВМ "Аргон" появились в 1968 г. В частности, ЦВМ «Аргон-11С» работала на борту космического аппарата «Зонд», облетевшего Луну и сфотографировавших ее обратную сторону.

В 1968 г. НИЭМ входит в Научно-исследовательский центр электронной вычислительной техники (НИЦЭВТ), созданный для проектирования ЕС ЭВМ. Директором НИЦЭВТ становится С.А.Крутовских, его заместителем - В.К.Левин, разрабатывавший аван-проект ЕС ЭВМ.

Дискуссии о концепции и прототипах ЕС ЭВМ ведутся на протяжении 1967-69 годов и в конце концов побеждают сторонники (С.А.Крутовских, В.К.Левин, В.В.Пржиялковскй, М.Р.Шура-Бура) построения ЕС ЭВМ на базе IBM-360. Против такого варианта активно выступали В.М.Глушков, С.А.Лебедев, Б.И.Рамеев. М.К.Сулим, но безуспешно. Дискуссии о правильности принятых в то время решений продолжаются до сих пор. Противники ориентации на IBM-360 эти решения считают роковыми, затормозившими развитие отечественной вычислительной техники на многие годы. Б.И.Рамеев в знак протеста покинул НИЦЭВТ и в дальнейшем работал в ГКНТ.

В СКБ-245 и далее в НИИсчетмаш, руководимом М.К.Сулимом, совместно с ИТМиВТ разрабатывается машина М-20, а затем ее вариант на транзисторной элементной базе М-222.

Михаил Кириллович Сулим - один из видных разработчиков и организатор производства вычислительной техники в СССР.

Участник Великой отечественной войны М.К.Сулим окончил Киевский политехнический институт в 1951 г. Работал в СКБ-245, в НИИсчетмаш. Участвовал в создании М-20. После М-20 разрабатывались М-220 и М-222, главным конструктором этих машин был М.К.Сулим.

С 1959 г. Михаил Кириллович занимается руководящей работой в Госкомитете по радиоэлектронике, и далее в Министерстве радиопромышленности, где он дошел до должности заместителя министра. Курировал работу НИИ, КБ и заводов, проектировавших и выпускавших вычислительную технику.

После принятия решения о создании ЕС ЭВМ на базе IBM-360 в знак протеста М.К.Сулим покинул пост заместителя министра и вернулся на работу в НИИсчетмаш в качестве директора.

Нельзя не упомянуть специализированные ЭВМ, разработанные в ЦНИИ «Агат» под руководством Я.А.Хетагурова. Ярослав Афанасьевич родился в 1926 г., окончил МВТУ им. Н.Э.Баумана. В 1962 г. появляется первая отечественная подвижная (в автоприцепе) полупроводниковая машина "Курс-1", предназначенная для работы в системе

противовоздушной обороны страны. Эта машина серийно изготавливалась на заводах Минрадиопрома вплоть до 1987 г. В интересах Военно-морского флота страны в «Агат» был создан ряд корабельных цифровых вычислительных систем, в том числе обеспечивавших стрельбу стратегического ракетного комплекса с подводной лодки.

Среди оригинальных разработок, выполненных в 60-е годы, следует назвать машины в остаточных классах Т-340А (1963 г.) и К-340А (1966 г.), созданные в НИИ-37 под руководством Д.И.Юдицкого и И.Я.Акушского.

Одним из крупных центров компьютерной промышленности в СССР, начиная с 60-х годов был Минск, где созданы завод ЭВМ и СКБ завода, позднее ставшее НИИ ЭВМ. Возглавил СКБ в 1964 г. Георгий Павлович Лопато (1924-2003). Его детищем является серия ЭВМ "Минск" (первая из машин серии «Минск-1» создана в 1960 г.). Под его руководством по заказу Минобороны разработан ряд мобильных вычислительных машин, совместимых с машинами ЕС ЭВМ.

В 1961 г. в Ленинграде на базе лаборатории, в которой работали приехавшие из-за рубежа Филипп Георгиевич Старос и Иозеф Вениаминович Берг, было создано конструкторское бюро КБ-2. В 1962 г. в КБ-2 была закончена разработка управляющей ЭВМ УМ1-НХ, нашедшей широкое применение в народном хозяйстве, а в 1964 г. - микроминиатюрная ЭВМ УМ-2, ориентированная на применение в аэрокосмических объектах. Но наиболее значительным результатом деятельности Ф.Г.Староса является его вклад в создание Научного центра микроэлектроники в Зеленограде, где некоторое время он работал главным инженером Центра и где использовались результаты разработки интегральных схем, полученные в КБ-2.

ЕС и СМ ЭВМ. Начиная с 1969 г., радиоэлектронная промышленность СССР переключилась на производство преимущественно машин ЕС и СМ ЭВМ.

Уже после появления первых ЭВМ стала очевидной целесообразность перехода к построению единого ряда машин разной производительности (ЕС ЭВМ), но согласованных по системе команд, операционным системам и требованиям к определенным характеристикам внешних устройств. С этой целью в 1967 г. в Москве был создан Научно-исследовательский центр электронной вычислительной техники (НИЦЭВТ).

В СССР дискуссии относительно проекта ЕС ЭВМ велись во второй половине 60-х годов прошлого века. Обсуждались две альтернативы построения единого ряда: 1) на основе развития отечественного и западноевропейского опыта; 2) на основе американской серии машин IBM-360.

Первый вариант позволял продолжить развитие отечественного научного и инженерного потенциала с шансами сохранения конкурентоспособности отечественных ЭВМ, поскольку к этому времени мы имели одну из лучших машин в мире БЭСМ-6 и серию машин "Урал". При этом предполагалось взаимовыгодное сотрудничество с английскими и немецкими фирмами, разрабатывавшими ЭВМ, поскольку эти фирмы тоже стремились к сотрудничеству с советскими специалистами.

Положительной стороной второго варианта была возможность использования программного обеспечения, уже созданного для IBM-360. Ведь IBM начала выпуск серии компьютеров IBM-360 еще в 1964 г. Для английских и немецких ЭВМ столь объемных наработок не было. Сторонники с вариантом базирования на IBM-360 справедливо считали, что необходимо существенно расширить применение ЭВМ в

народном хозяйстве, а этого без богатого программного обеспечения не сделаешь. Причем имелось в виду наличие программного обеспечения не только на данный момент, но и в перспективе, а у нас в стране в то время (по данным академика А.А.Дородницына) было приблизительно 1500 квалифицированных программистов по сравнению с 50000 в США. Кроме того, из ориентации на IBM-360 вытекало лишь требование совместимости с системой команд IBM-360 и еще не означало слепого копирования чужих решений.

Было принято решение в пользу второго варианта, и НИЦЭВТ становится головной организацией по программе ЕС ЭВМ [15].

Аван-проект ЕС ЭВМ разрабатывало Конструкторское бюро промышленной автоматики (КБПА) во главе с В.К.Левиным, а головной организацией по вопросам математического обеспечения стал Институт прикладной математики, где эти работы возглавляли М. Р. Шура-Бура и В. С. Штаркман.

Михаил Романович Шура-Бура родился в 1918 г., известен как крупнейший специалист в области вычислительной техники и программирования.

В 1971 г. прошла совместные испытания первая машина Единой системы ЕС-1020, разработанная Минским НИИ ЭВМ (гл. конструктор В. В. Пржиялковский). В 1972 г. Ереванским НИИММ сдана ЕС-1030 (гл. конструктор М. А. Семерджян). В 1973 г. в ГДР под руководством гл. конструктора М. Гюнтера создана ЕС-1040. В НИЦЭВТ закончена разработка старших моделей: в 1973 г. ЕС-1050, в 1977 г. – ЕС-1060 (гл. конструктор обеих моделей В.С.Антонов), в 1984 г – ЕС-1066 (гл. конструктор Ю.С.Ломов). Генеральными конструкторами ЕС ЭВМ в этот период были С.А.Крутовских, (1968-1970 гг.), А.М.Ларионов (1970-1977 гг.), В.В.Пржиялковский (1977-1990 гг.), одновременно являвшиеся директорами НИЦЭВТ.

Для производства машин ЕС ЭВМ были задействованы заводы в Минске, Ереване, Казани, Пензе, Вильнюсе и в странах СЭВ.

К 1979 г. доля ЕС ЭВМ в парке ЭВМ страны составляла 72%. В серии ЕС ЭВМ наиболее массовыми были машины ЕС-1022 (к 1989 г. было выпущено около 3400 машин), ЕС-1033 (1405), ЕС-1035 (1872), ЕС-1045 (1069). Высокопроизводительных машин ЕС-1055, ЕС-1060 и ЕС-1061 было произведено по несколько сотен. Всего за 20 лет промышленностью были поставлены для народного хозяйства и обороны страны более 16 тыс. вычислительных комплексов ЕС ЭВМ.
Однако по своему техническому уровню эти машины значительно отстают от американских машин того же времени.

В начале 1974 г. в СССР было принято решение о создании семейств не только больших, но и малых ЭВМ с ориентацией на архитектуру машин PDP-11 американской компании DEC. Семейство малых машин получило название СМ ЭВМ. В развитии СМ ЭВМ значительную роль сыграл Институт электронных управляющих машин под руководством Б.Н.Наумова.

Борис Николаевич Наумов (1927-1988) в 1950 г. окончил МЭИ по специальности "Автоматическое управление". В 1950-1967 гг. Б.Н.Наумов работал в Институте автоматики и телемеханики (ИАТ). В 1958—1959 г. был в командировке в Массачусетском технологическом институте (США), где встречался с Н.Винером. В 1967 г. возглавил ИНЭУМ (институт электронных управляющих машин) и стал главным конструктором АСВТ-М (Агрегатная система средств вычислительной техники на микроэлектронной базе). Уже в 1970 г. были созданы первые в стране управляющие

вычислительные комплексы третьего поколения. При этом начали использоваться методы совмещенного (параллельного) проектирования.

В период 1974-1984 гг. Б. Н. Наумов руководил разработкой системы малых ЭВМ (СМ ЭВМ) в качестве Генерального конструктора.

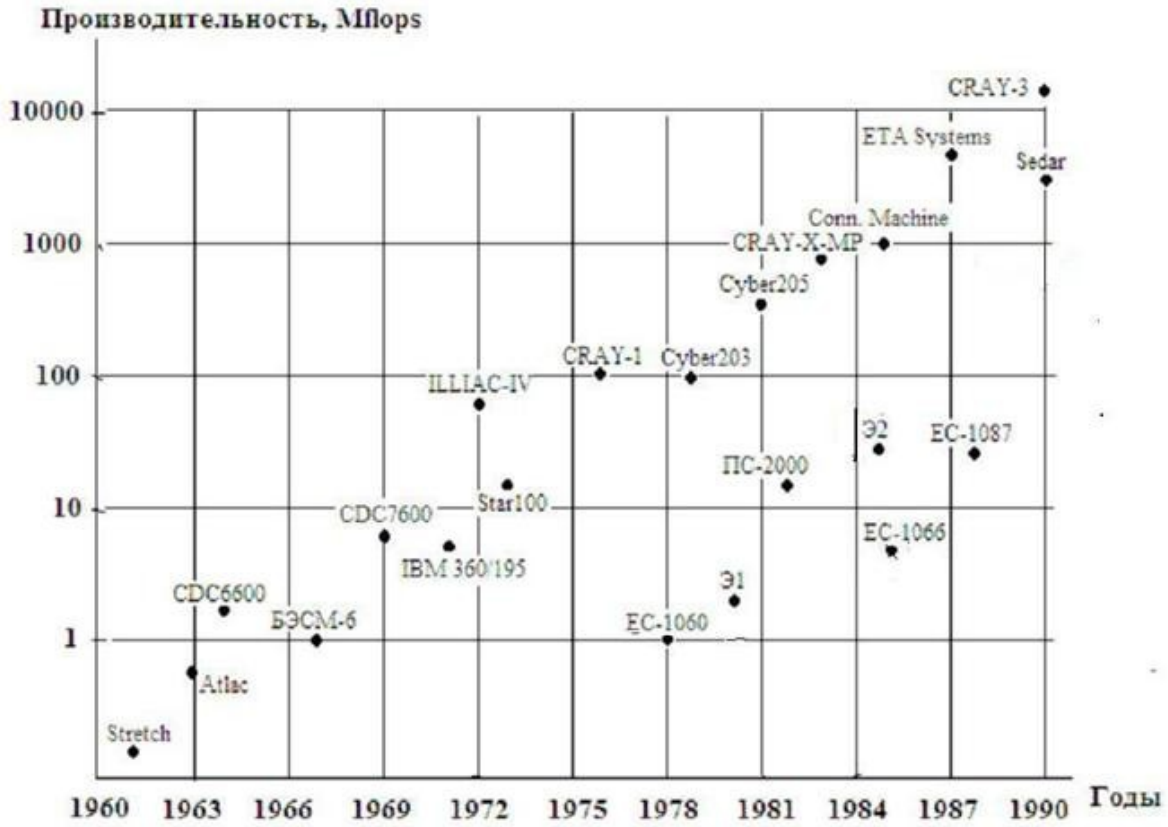
Одной машиной из серии СМ ЭВМ был компьютер СМ-1420, предназначенный для работы в составе АСУТП, систем сбора, подготовки и обработки данных, систем автоматизации научных экспериментов и т.п. Среднее быстродействие этой машины 0,30 Mips или 0,23 Mflops. Оперативная память 248 К.

Б. Н. Наумов был одним из инициаторов организации в составе АН СССР Отделения информатики, вычислительной техники и автоматизации, созданного в 1983 г. Б.Н.Наумов возглавил Институт проблем информатики АН СССР (ИПИАН), организованный по его инициативе.

СМ-1420

Для СМ ЭВМ были приняты стандарты «де-факто» архитектур малых ЭВМ, наиболее распространенных в мире, предложены интерфейсы, обеспечивающие использование общей для всех моделей номенклатуры периферийных устройств и устройств связи с объектом. Разработанные под руководством Б.Н.Наумова принципы и стандарты СМ ЭВМ, охватывающие в комплексе все аспекты унификации элементов, узлов и устройств, конструкций, рядов моделей ЭВМ, средств программирования, учитывали технологию и производственные возможности отечественной промышленности и обеспечили возможность организации крупносерийного производства.

На базе СМ ЭВМ был реализован ряд специализированных комплексов. Например, комплекс СМ-4 вместе с Фурье-процессором использовался для обработки радиолокационных изображений поверхности Венеры, что позволило справиться с уникальной по сложности задачей с помощью мини-ЭВМ вместо суперЭВМ



К сожалению, именно с конца 60-х – начала 70-х годов, когда принято решение о построении ЕС ЭВМ на базе IBM-360, начинается отставание отечественной вычислительной техники от зарубежной. Среди причин можно назвать трудности становления НИЦЭВТа, как слаженно работающего коллектива, так как он собран из групп разработчиков нескольких организаций. Возможно, наличие прототипа IBM-360 так или иначе сковывало творческий потенциал разработчиков, направляя его на выяснение чужих решений. Во всяком случае, из рис. 1, на котором показано, как в 60-80-е годы изменялось быстродействие вычислительных систем, видно, что рост производительности ЭВМ в мире в целом подчинялся закону Г.Мура, а в отношении отечественных ЭВМ он был нарушен. Если БЭСМ-6 находилась в общем потоке роста производительности, практически не уступая лучшим зарубежным ЭВМ, то равноценную по производительности ЕС-1060 удалось получить только через 11 лет, когда американские разработчики ЭВМ ушли далеко вперед.

Но является ли решение о построении ЕС ЭВМ именно на базе IBM-360 причиной нашего отставания? Начиная с середины 70-х годов, когда в мире произошел переход к ЭВМ четвертого поколения, основным фактором нашего отставания следует все-таки считать отсутствие элементной базы, сопоставимой с зарубежными БИС и СБИС. Об этом свидетельствует тот факт, что линия Эльбрусов (машины Э1 и Э2) на рис. 1 также находится ниже общемировой тенденции роста производительности суперкомпьютеров.

А экономическая разруха 90-х годов усугубила ситуацию, отбросив Россию в число стран, отстающих не только от США, но также от многих стран Европы, Азии и даже Африки.

Производство ЕС ЭВМ в России окончательно прекратилось в 1995 г.

В последние годы в НИЦЭВТ, потерявшем значительную часть своего потенциала, разрабатываются вычислительные кластеры и серверы на базе современных коммерчески доступных компонентов.

Беда нашей вычислительной техники - не только значительное отставание само по себе. Как сказал в конце 80-х академик А.П.Ершов: "Мы не отстаем – мы идем не туда".

Отечественные суперкомпьютеры. Исследования по многопроцессорным вычислительным системам в СССР были начаты в начале 60-х. Возможность построения суперкомпьютеров на принципах параллельного выполнения операций в однородных вычислительных средах была показана Э. В. Евреиновым и Ю. Г. Косаревым в Новосибирске в 1962 г. Работы, проводимые в Таганрогском радиотехническом институте под руководством А.В.Каляева (1922-2004), впоследствии ставшего академиком РАН, привели к созданию ряда многопроцессорных специализированных ЭВМ, первой из них в 1964 г. была создана цифровая интегрирующая машина Метеор-3.

Работы по использованию параллелизма в универсальных ЭВМ после принятия решения о построении ЕС ЭВМ на базе IBM-360 были продолжены в Москве. Над созданием суперЭВМ трудились ученики С.А.Лебедева – В.А.Мельников, В.С.Бурцев, Б.А.Бабаян.

Владимир Андреевич Мельников (1928-1993), выпускник МЭИ, участвовал в разработке БЭСМ, БЭСМ-2, БЭСМ-6, АС-6. В 80-е годы В. А. Мельников создает Институт проблем кибернетики АН СССР и становится его директором. Здесь он руководит разработкой векторно-конвейерной суперЭВМ «Электроника ССБИС», близкой по своей архитектуре к американской суперЭВМ Cray-1. Изготовление этой ЭВМ происходит на Калининградском заводе вычислительных машин.

В.С.Бурцев продолжает линию Эльбрусов в ИТМиВТ.

Всеволод Сергеевич Бурцев родился в 1927 г. Окончил Московский Энергетический Институт в 1951 г., в 1973-1984 годах возглавлял Институт Точной Механики и Вычислительной Техники АН СССР, в дальнейшем был директором ВЦКП РАН и Института Высокопроизводительных Вычислительных Систем. В 1992 г. стал действительным членом РАН.

В 1979 г. появляется Эльбрус-1 - компьютер на основе суперскалярного RISC-процессора, разработанный в ИТМиВТ, генеральный конструктор В.С.Бурцев. В 1984 гг. под его руководством создан 10-процессорный суперкомпьютер Эльбрус-2, который использовался в Российской противоракетной системе, ЦУПе, Арзамасе-16 и Челябинске-70. .

Проект 16-процессорного компьютера Эльбрус-3 производительностью 125 млн операций в секунду с большой локальной оперативной памятью для каждого процессора (16 Мбайт) и глобальной общей для всех процессоров памятью (2 Гбайт) появился в 1985 г. Ключевой фигурой в его создании был Борис Арташесович Бабаян, окончивший Московский физико-технический институт (в 1957 г.). Параллельно с развитием Эльбрусов в 80-е годы разрабатывались матричные процессоры ПС-2000 и ПС-3000. Однако громоздкие Эльбрусы, несмотря на использование в них ряда интересных архитектурных решений, проигрывали зарубежным суперкомпьютерам из-за несовершенной элементной базы.

В 1986 году вышло постановление правительства СССР о создании вычислительного комплекса «Эльбрус-90 микро». Обязательным условием было использование в новой разработке только отечественных решений, элементной базы и программного обеспечения. К 1990 году микропроцессор для «Эльбрус-90 микро» был спроектирован, началась подготовка к его изготовлению в Зеленограде. Но в стране начались политические и экономические преобразования, обрушившие электронную промышленность. Большинство разработчиков из ИТМиВТ ушло, часть из них оказалась в компании МЦСТ, созданной Б.А.Бабаяном. Именно в МЦСТ (научный руководитель Б.А.Бабаян, генеральный директор Александр Ким), входящей в группу компаний Эльбрус, были продолжены работы над компьютерами семейства Эльбрус.

В конце 1997 года были завершены заводские, а в 1998-м - государственные испытания «Эльбруса-90 микро», утверждена документация для серийного производства, изготовлена опытная партия. «Эльбрус-90 микро» отличается от предыдущих Эльбрусов несравненно меньшими габаритами и большей надежностью.

Дальнейшие разработки МЦСТ - микропроцессор E2k, создаваемый по архитектуре EPIC (командные слова по 512 бит) и 0,13 мкм технологии стандартных блоков, и суперЭВМ Эльбрус-3М с производительностью 8 млрд операций/с на один процессор. Эти решения полностью отечественные, хотя заказы на производство микропроцессоров E2k размещаются в Израиле и на Тайване.

В 2004 г. коллектив разработчиков Эльбрусов во главе с Б.А.Бабаяном перешел на работу в компанию Intel.

Современные отечественные суперЭВМ строятся на зарубежной элементной базе.

В 2002 г. в список 500 наиболее производительных компьютеров мира (Top500) впервые вошел российский суперкомпьютер, заняв 74-е место. Это суперкомпьютер МВС1000М, установленный в Межведомственном суперкомпьютерном центре (создан в 1996 году совместным решением Российской академии наук, Министерством науки и технологии, Министерством образования и Российским фондом фундаментальных исследований) и имеющий производительность 735 Gflops. Его разработка велась под руководством В.К.Левина.

Владимир Константинович Левин родился в 1929 г., окончил Московский энергетический институт в 1950 г. Далее работал в Конструкторском бюро промышленной автоматики (КБПА) Министерства радиопромышленности. Был в 1958–1965 гг. заместителем главного конструктора первой отечественной полупроводниковой высокопроизводительной вычислительной машины общего назначения «Весна». С 1968 по 1976 гг. работал главным инженером НИЦЭВТ, затем вернулся в КБПА, переименованное в НИИ «Квант», для разработки высокоскоростных вычислителей. Под его руководством созданы многопроцессорные вычислительные системы (МВС), упоминаемые в списках Top500.

В состав МВС1000М входят 5 вычислительных узлов, один управляющий узел, коммутирующая сеть Myrinet. Суперкомпьютер построен на процессорах Alpha, число процессоров 768. Объем оперативной памяти системы - 768 Гбайт. Система работает под управлением операционной системы Red Hat Linux 6.2, поддерживающей многопроцессорные системы.

МВС-1000М

В списке Top500 2004 года на 210 месте значится новый российский компьютер МВС5000БМ производительностью 1,4 Tflops, выполненный как BladeServer на 336 микропроцессорах PowerPC 1,6 ГГц, коммутирующая система Myrinet.

Отрадно, что в этом списке на 98-м месте появился установленный в Белоруссии компьютер СКИФ К-1000 с производительностью в 2 Tflops, в создании которого участвовали около 20 российских и белорусских предприятий, включая Институт программных систем РАН. Он выполнен на микропроцессорах Opteron 2,2 ГГц, коммутирующая система построена на основе технологии Infiniband.

Основные вехи в истории телекоммуникаций и сетей.

Еще до появления первых компьютерных сетей были созданы теоретические основы телекоммуникаций и электросвязи.

В 1948 г. Клод Шеннон (Claude Shannon) опубликовал работу «Математическая теория связи», заложившую фундамент современной теории связи. В ней он представил свою унифицированную теорию передачи и обработки информации, предложил оценку количества информации.

Теория кодирования ведет свою историю с работы Ричарда Хэмминга (1950 г.), в которой им предложен блочный код, корректирующий одиночные ошибки, возникающие при передаче сообщений.

В 1961 г. Леонард Клейнрок (Leonard Kleinrock) опубликовал работу, в которой он предложил выполнять передачу данных с помощью коммутации пакетов. Смысл технологии коммутации пакетов заключается в разделении сообщения на части (пакеты), передаче пакетов по сети и сборке сообщения в узле назначения. Работая в Калифорнийском университете в Лос-Анджелесе, Л.Клейнрок внес значительный вклад в создание первой сети ARPANET.

В 1962 г. концепцию компьютерной сети предлагает также Джон Ликлайдер (J.C.R.Licklider) из Массачусетского технологического института (MIT). В своей работе "Galactic Network" он говорит о возможности существования в будущем глобальной компьютерной связи между людьми, имеющими мгновенный доступ к программам и базам данных из любой точки земного шара, т.е. предсказывает появление сети Internet.

В 1965 г Лоренс Робертс и Томас Мерилл впервые соединили два удаленных компьютера с помощью низкоскоростных телефонных коммутируемых линий, экспериментально доказав возможность построения территориальных сетей. А сеть ARPANET, являющаяся прообразом Интернет, создана в 1969 г., благодаря совместным

усилиям Калифорнийского университета и специалистов из агентства ARPA, которое возглавил Д.Ликлайдер.

Агентство ARPA (Агентство передовых исследовательских проектов или Advanced Research Projects Agency) было создано для поддержки науки и образования в США сразу после запуска в СССР первого искусственного спутника Земли. Перед ARPA поставлена задача - ликвидировать отставание США в космических исследованиях. Агентство располагало несколькими миллиардами долларов, которые оно распределяло между университетами и компаниями, ведущими разработки компьютерных сетей. Под руководством Д.Ликлайдера в ARPA велись работы, которые и привели к созданию ARPANET путем создания связи между компьютерами Агентства и Стенфордского университета. Первая попытка передачи данных между четырьмя удаленными компьютерами была предпринята в 1969 г., а в начале 70-х годов ARPANET работала с пиковой скоростью 0,25 Мбит/с, обслуживая около 200 пользователей.

Но еще раньше (1967 г.) первую в мире локальную вычислительную сеть (ЛВС) создает Дональд Дэвис (Donald Davies) в Национальной физической лаборатории Великобритании (British National Physics Laboratory). Д.Дэвис в дальнейшем занимается проблемами защиты информации.

В 1968 г. в Швеции Олаф Содерблюм из компании IBM разработал локальную сеть Token Ring. В том же году Министерство обороны США выпускает первый в мире стандарт на сетевые технологии MIL-STD-1553, посвященный локальным вычислительным сетям. В.Чу (W. W. Chu) ввел термин “Asynchronous Time Division Multiplexing” — так зарождается идея технологии ATM.

Очевидно, что для взаимодействия сетей они должны обладать свойством открытости. Открытость обеспечивается прежде всего стандартизацией протоколов. Важную роль в развитии стандартизации в области сетевых технологий играет Институт инженеров по электротехнике и электронике (IEEE). История IEEE начинается с появления в 1884 г. Американского института инженеров по электротехнике. В 1912 г. создается Институт радиоинженеров (The Institute of Radio Engineers), в нем организуется комитет стандартов. В 1963 г. эти институты объединились, породив IEEE.

В 1970 г. на Гавайских островах Норман Абрамсон (Norman Abramson) создал сеть Aloha — прообраз Ethernet и RadioEthernet. Это была первая в мире пакетная радиосеть, использовавшая случайный метод доступа к среде передачи данных - пакеты передавались в эфир, когда в этом возникала необходимость. Если через какое-то время возвращалось посланное таким же простым методом подтверждение получения, то сообщение считалось доставленным. Если подтверждение не приходило, следовала повторная попытка передачи.

В 1972 г. Рэй Томлисон (Ray Tomlison) из компании BBN (США) разработал систему электронной почты. С тех пор более чем на десять лет электронная почта была крупнейшим сетевым приложением и остается важнейшим сервисом после Web.

1973 г. отмечен появлением мобильной телефонной связи.

Этот год можно считать датой начала работ над Ethernet - Роберт Меткалф (фирма Xerox) подает записку с предложением создать Ethernet – одну из первых в мире локальных вычислительных сетей. Слово Ethernet происходит от "ether" - эфир, оно использовано, поскольку одним из предшественников Ethernet была радиосеть PRNET. Проект Ethernet реализован Р.Меткалфом в 1976 г.

Роберт Меткалф родился в 1946 г. Учился в Массачусетском технологическом институте. Участвовал в разработке сети ARPANET. Сеть Ethernet он создал, выполняя задание фирмы по перекачке данных с компьютера на лазерный принтер со скоростью, существенно более высокой чем допускает RS-232. Впоследствии Меткалф основал известную телекоммуникационную компанию 3Com.

В 1974 г. группа Internet Network Working Group (INWG), руководимая Винтоном Серфом, представила проект универсального протокола передачи данных и объединения сетей - TCP/IP. Далее проект дорабатывался специалистами ARPA и в 1979 г. стек протоколов TCP/IP был сформирован.

В 1975 г. фирмой DEC создается сеть Decnet, развивавшаяся вплоть до 1990 г.

В 1977 г. достигнуто объединение компьютеров в сеть на общей платформе. Тем самым ARPANET преобразуется в Internet.

В 1978 г. Международная организация стандартизации разработала семиуровневую модель открытой сетевой архитектуры.

В 1979 г. три ведущие фирмы — Xerox, DEC и Intel — объединили свои усилия, чтобы стандартизовать Ethernet. Произошло это при посредничестве Р.Меткалфа, который считает это объединение даже более важной своей заслугой, чем изобретение самой Ethernet.

В том же году Американский национальный институт стандартов (ANSI) сформировал группу специалистов для разработки высокоскоростного канала обмена данными. Результатом разработки стала высокоскоростная (по тем временам) локальная вычислительная сеть FDDI.

В 1981 г. создана аналоговая система сотовой связи NMT-450.

В 1981 г. начинает функционировать глобальная сеть Bitnet (Because It's Time Network), объединяющая преимущественно университеты и научные центры.

В 1982 г. Европейская конференция администраций почт и электросвязи (CEPT), объединяющая администрации связи 26 стран, создала специальную группу Groupe Special Mobile (GSM). Аббревиатура наименования группы и дала название новому стандарту. Тем не менее, позднее в связи с широким распространением этого стандарта во всем мире, GSM стали расшифровывать как «Global System for Mobile Communications».

В 1983 г. происходит перевод Internet (точнее, ARPANET) на стек протоколов TCP/IP, состоявшийся 1 января. Переход тщательно планировался всеми заинтересованными сторонами в течение нескольких предшествующих лет и прошел на удивление гладко (аналогично переходу в третье тысячелетие - решению проблемы 2000 г.) В том же году введена доменная система имен DNS.

В 1986 г. на основе технологии ARPANET была создана NSFNET (the National Science Foundation NETwork - сеть национального научного фонда США), в создании которой приняли непосредственное участие NASA и Министерство энергетики США. Задачей этой сети было предоставление научной общественности США доступа к суперкомпьютерам, а также создание основной опорной межрегиональной магистрали (Backbone) с базовой скоростью 56 Кбит/с. Предложенная магистральная сеть имела

иерархическую структуру - локальные подсети соединялись через один из своих узлов с суперкомпьютерным центром, центры были связаны друг с другом.

Существовавшие магистральные сети нуждались в повышении пропускной способности и в унификации протоколов для совместимости разных сетей. Ответом на эти потребности было создание в 1987 г. канала Т-1 со скоростью 1536 Кбит/с. В 1990 г. появляется магистраль Т-3 с пропускной способностью 45 Мбит/с.

Известно, что термин “гипертекст” был впервые предложен Тедом Нельсоном (Theodor Nelson) в 1965 г., а первые работающие гипертекстовые системы создали в 1967 г. Энди Ван Дам (Andy van Dam) и в 1968 г. - изобретатель «мыши» Дуг Энгельбарт (Douglas Englebart). Иногда их называют отцами гипертекста.

Однако первые признаки гипертекста можно найти еще в работах Ванневара Буша. Именно он в 1945 г. в проекте машины Memex предложил механизм перекрестных ссылок, аналогичный используемому в современном гипертексте. Ссылки предлагалось реализовать с помощью вспомогательных полей в теле документов. Этот механизм направлен на создание из отдельных документов виртуальных книг в соответствии с потребностями пользователей.

История языков разметки началась в 60-е годы, когда сотрудники IBM Ч.Гольтфарб, Э.Мошер и Р.Лори разработали язык GML (General Markup Language) для переноса документов между разными вычислительными устройствами. Уже в 80-е годы этот язык получил название SGML в стандарте ISO 8879. Язык SGML по-прежнему рассматривается как основной язык разметки в современных интерактивных технических руководствах по эксплуатации сложной техники.

В 1990 г. британец Тим Бернерс-Ли вместе с Робертом Киллиау (R.Cailliau) в Европейской физической лаборатории (CERN) разработал язык гипертекстовой разметки HTML, предложил протокол World Wide Web (WWW) для специальной сети, объединяющей физиков мира, и разработал первый Web-браузер. Бернерсу-Ли принадлежит приоритет в трех важнейших компонентах Web: определение спецификаций URL (Universal Resource Locator), протокол HTTP (HyperText Transfer Protocol) и собственно язык разметки HTML (HyperText Markup Language).

Среди предшественников Т.Бернерса-Ли заметное место занимает Тед Нельсон. Ему принадлежит грандиозный, но так и не осуществленный проект глобальной гипертекстовой базы письменных документов Xanadu (1981 г.).

В 1993 г. Марк Андреесен (M.Andreessen), работавший в Университете штата Иллинойс (США), написал программу первого графического сетевого браузера Мозаик, а в 1994 г. создана поисковая система "Yahoo!". Именно с момента выпуска Мозаик началось триумфальное шествие WWW по планете. В дальнейшем М.Андреесен создал и возглавил компанию Netscape Communications.

В 1994 г. образовался консорциум W3C (W3 Consortium), который стал заниматься вопросами стандартизации в мире Интернета. Прежде всего была разработана спецификация HTML 2.0. Далее была добавлена система CSS (Cascading Style Sheets).

В 1996 г. на базе SGML и HTML создан язык XML, который вместе с HTML составляет основу для разработки баз гипертекстовых документов в Internet.

День 10 февраля 2004 г. принято считать днем рождения семантической сети (Semantic Web), в этот день консорциум W3C, возглавляемый Т.Бернерс Ли, утвердил спецификацию языка сетевых онтологий OWL (Web Ontology Language).

В 90-е годы получили развитие высокоскоростные локальные сети. В 1993 г. была предпринята попытка объединить технологии Ethernet и Token Ring, в результате появилась сеть 100VG-AnyLAN с информационной скоростью в 100 Мбит/с. Но более значимыми были результаты развития собственно Ethernet: в 1995 г. принят стандарт IEEE 802.3u на сеть Fast Ethernet, в том же году начаты работы над Gigabit Ethernet, а в 2002 г. стандартизована технология 10GE 10-гигабитной Ethernet.

В области подвижной радиосвязи общего пользования одним из важных событий было появление в 1995 г. стандарта IS-95 - первого стандарта сотовой связи, посвященного технологии кодового разделения каналов CDMA.

В СССР одной из первых ЛВС стала ЛВС в ФИАН им. П. Н. Лебедева (Москва), разработанная в 1975 г.

История российского Интернета не слишком богата событиями. В Советский Союз Интернет пришел в 1990 г. - 1 августа компания Релком (компьютерная сеть того же названия была создана на базе Курчатовского института атомной энергии) объединила несколько своих сетей на территории СССР. Вскоре сеть Релком была подсоединена к европейской сети EUNET. В 1993 г. сеть EUnet/Relcom была официально подключена к Интернету и был зарегистрирован домен RU, что и следует считать началом присутствия России в Интернете.

В 90-е годы создан ряд других российских территориальных сетей Runnet, Роспак и др., используются как волоконно-оптические, так и спутниковые магистральные каналы передачи данных.

Элементная база компьютеров.

Основными вехами развития электронной промышленности в мире стало создание транзистора (1948), интегральных схем (1958) и микропроцессора (1971), ознаменовавшими второе, третье и четвертое поколения компьютеров.

Изобретателем транзистора является американский физик. Уильям Шокли (Chockley) (1910-1989), В 1947 г. ему вместе с Дж. Бардином и У. Браттейном удалось получить точечный транзистор, а в 1951 г. первый плоскостной германиевый транзистор. В 1954 г. Гордон Тил разработал первый кремниевый биполярный транзистор.

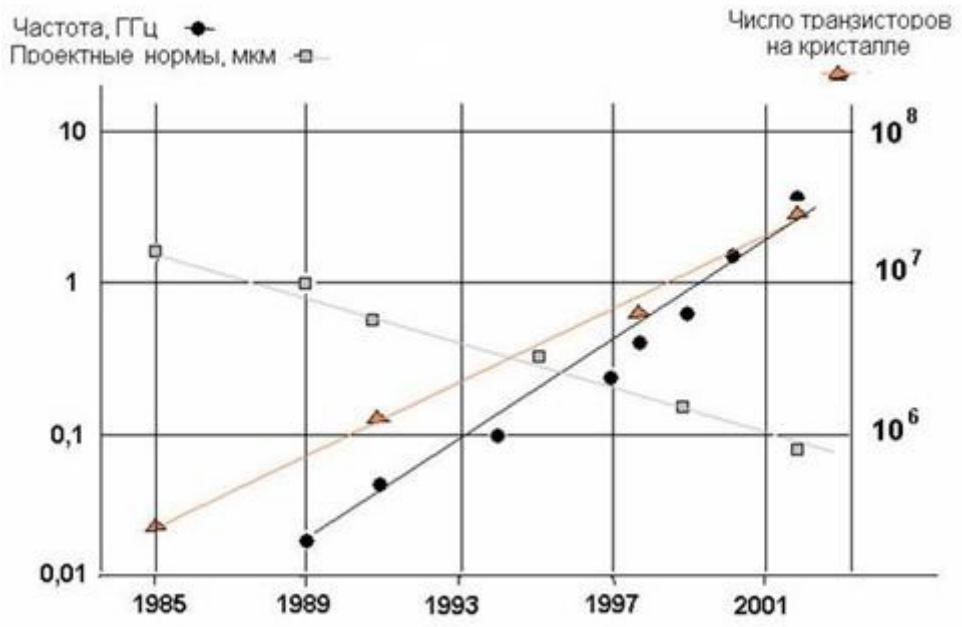
Изобретателями первых интегральных схем (в виде системы взаимосвязанных транзисторов на единой кремниевой пластине) были Роберт Нойс (Robert Noyce) (1927-1990) из компании Fairchild Semiconductor и независимо от него Джек Килби (Kilby) из компании Texas Instruments. Более удачными были признаны схемы Нойса.

В 1968 г. Р.Нойс и Г.Мур основали компанию Intel (сокращение от Integrated Electronics), а в следующем году - Дж. Сандерс создает компанию AMD (Advanced Micro Devices). Этим компаниям суждено на протяжении многих лет конкурировать на рынке микропроцессоров.

Первый микропроцессор 4004 разработан и изготовлен в 1971 г в компании Intel с помощью инженера из Стэнфорда Т. Хоффа. В дальнейшем наблюдается неуклонное

повышение тактовой частоты микропроцессоров, характеризующей их быстродействие, в соответствии с законом, сформулированным Г.Муром. Увеличение частоты основано на технологических достижениях, которые непосредственно выражаются в уменьшении проектных норм и, следовательно, в увеличении степени интеграции, а также в росте размеров кремниевой пластины, в которой формируются микропроцессорные СБИС.

Микросхема 4004, изготовленная в 1971 г., была четырехразрядной и включала немногим более двух тысяч транзисторов. Первый персональный компьютер был создан компанией IBM на основе восьмиразрядного микропроцессора 8088. Первый 16-разрядный микропроцессор, открывший серию 80X86, появился на рынке в 1978 г. Он был изготовлен по технологии с проектными норами в 3 мкм, включал 29 тыс. транзисторов, работал на частоте 5 МГц. Переход к 32-разрядным процессорам произошел в 1985 г., начиная с моделей серии 80386, а эпоха 64-разрядных процессоров с CISC-архитектурой началась в 2000 г. с создания процессора Itanium.



Основные вехи увеличения тактовой частоты микропроцессоров Intel, снижения минимального топологического размера (проектных норм) в технологиях их изготовления и роста числа транзисторов на кристалле показаны на рис. 2.

Использование 64-разрядной архитектуры для процессоров RISC довольно типично, так 64-разрядными являются процессоры Sun UltraSPARC, Alpha, MIPS R1x000, IBM Power, HP PA-8x00. Например, микропроцессор Alpha был разработан корпорацией Digital Equipment на рубеже 80-90-х годов. Этот 64-разрядный RISC-процессор суперскалярной архитектуры предназначен для высокоскоростных вычислений. 64-разрядный микропроцессор Power был реализован в 1995 году в компьютере IBM AS/400.

В Intel используется CISC-подобная архитектура и переход от 32-разрядной архитектуры IA-32 к 64-разрядной IA-64 сопряжен с определенными трудностями. Первый микропроцессор архитектуры IA-64, появившийся в 2000 г., назван Itanium, уже созданы следующие версии микропроцессоров Itanium. Созданием 64-разрядных CISC-подобными микропроцессорами занимается также компания AMD. Например, ее 64-разрядный микропроцессор Opteron с тактовой частотой 1,8 ГГц может выполнять как 64-, так и 32-разрядные приложения.

Кэширование памяти в ПК начато в 1989 г. с моделей i486. Кэш первого уровня разделен на кэши данных и команд, которые в первых моделях имели емкость по 8 Кбайт, в Pentium III - по 16 Кбайт. Кэш второго уровня L2 в моделях Pentium II имел емкость 512 Кбайт, работал с половинной частотой процессора и располагался на плате процессора. Но уже начиная с Pentium III Celeron, кэш L2 работает на частоте процессора и встроен в ядро. В новых процессорах кэш-память становится трехуровневой. Так емкость кэш-памяти первого, второго и третьего уровней, для Itanium составляет 96, 96, 4000 Кбайт соответственно, для Itanium II - 256, 256, 3000 Кбайт, причем у Itanium II все кэши интегрированы в кристалл микропроцессора. Кэши разных уровней в IA-64 обмениваются данными со скоростями от 13 до 32 Гбайт/с.

К числу ведущих компаний электронной промышленности в США относится также Motorola. В 1949 г. компания приступает к выпуску полупроводниковых приборов. В 1979 г. Motorola разрабатывает свой первый 16-битный микропроцессор 68000, а в 1984 г. - первый 32-битный микропроцессор MC68020, в котором содержится около 200000 транзисторов, обеспечивается доступ к памяти до 1 миллиарда бит.

Микропроцессор Power PC создается совместными усилиями компаний Apple Computer, IBM и Motorola в 1993 г., в нем используется RISC-технология. Представление о характеристиках RISC-процессоров можно получить из рассмотрения особенностей 64-разрядного процессора Power4, использовавшегося в мощных серверах IBM и компьютерах компании Apple. Это процессор суперскалярной архитектуры. На кристалле располагаются два процессорных ядра, кэш-память первого уровня емкостью 2´64 Кбайт и второго уровня объемом 1,5 Мбайт, коммутирующая матрица, обеспечивающая пропускную способность до 35 Гбайт/с. Внешний кэш третьего уровня имеет емкость до 32 Гбайт на один процессорный кристалл. Общее число транзисторов на кристалле - 170 миллионов. Микропроцессор изготовляется по КМДП технологии "кремний-на-изоляторе" с проектными нормами в 0,18 мкм и медными соединениями на кристалле Тактовые частоты начинаются с 1 - 2 ГГц, Процессор Power4 представлял собой объединение четырех кристаллов в один многокристальный модуль, т.е. микросхему Power4 можно рассматривать как компактную SMP-систему.

Первые работы в СССР по созданию интегральных схем проводили Ф.Г.Старос и И.В.Берг в КБ-2 в Ленинграде. В августе 1962 г. в СССР было принято решение о строительстве Научного центра микроэлектроники в г. Зеленограде и организации его филиалов в Киеве, Минске, Риге, Вильнюсе. До 1964 г. Ф Старос был главным инженером Центра.

Одним из ведущих предприятий Центра был НИИ молекулярной электроники (НИИМЭ), который в 1964 г. возглавил Камиль Ахметович Валиев. В 1967 г. рядом с НИИМЭ создается опытный завод Микрон. В этих организациях проектировались и изготовлялись многие серии отечественных микросхем. В настоящее время К.А.Валиев занимается исследованиями квантовых вычислений.

История программирования и алгоритмических языков.

Первым программистом в истории вычислительной техники считается леди Ада Лавлейс (1815-1852) - дочь лорда Байрона, отличавшаяся незаурядными математическими способностями. Сотрудничая с Ч.Бэббеджем, написала комментарии к статье о его машине, которые можно считать первым трудом по программированию. Она описала ряд примеров применения машины Бэббеджа, разработала программу вычисления на ней чисел Бернулли.

Одним из первых алгоритмических языков является Plankalkul, разработанный К.Цузе в 40-х годах прошлого века. В отличие от языков ассемблера Plankalkul не был привязан к архитектуре и наборам команд конкретного компьютера.

Первые алгоритмические языки эпохи ЭВМ относятся к середине 50-х годов. Это языки FORTRAN и ALGOL, ориентированные преимущественно на численные расчеты. Джону Бэкусу (J.Backus) принадлежит фундаментальный вклад в создание в 1954 г. языка Фортран (FORTRAN - FORmular TRANslation) и в разработку компилятора для IBM 705 (1958 г.). Коллектив под руководством Бэкуса разработал не только спецификацию языка, но и создал компилятор для него. Кроме того, Фортран был первым языком, для которого были созданы компиляторы для многих ЭВМ. Это обеспечило достаточно быструю и широкую распространенность языка,

Язык Алгол с первоначальным названием IAL и с более поздним ALGOL-58 был разработан в 1958 г. международным комитетом под руководством Питера Наура (P.Naur).

Новые версии языков FORTRAN II и ALGOL-60 начала 60-х годов были усовершенствованными вариантами своих предшественников. При описании языка Алгол Д.Бэкус использовал обозначения и средства, которые стали называться нотацией Бэкуса-Наура и синтаксическими диаграммами [16].

Язык Кобол (COBOL – Common Business Oriented Language), представленный в 1960 г., был предназначен для расчетов в сфере бизнеса.

Разработка языка PL/1 относится к 1964 г. Его разработчики из английского отделения IBM стремились в этом языке совместить возможности Алгола и Кобола.

Язык моделирования Simula-67 разработан в Норвегии Найгардом (Nygard) и Далом (Dahl). В нем уже используются многие черты объектно-ориентированного программирования.

Сложность разрабатываемых программ становилась все заметнее. Для преодоления возникающих трудностей требовалась разработка методологии программирования, превращающей стихийное программирование в науку создания программных систем. Заметным вкладом в создание технологий разработки программного обеспечения стала концепция структурного программирования [17], основанная на положениях блочно-иерархического подхода к программированию. Ее автором является голландский ученый Эдстер Дейкстра (1930-2002), внесший заметный вклад в разработку информационных технологий Ему принадлежат также определения стека, семафора, разработка ряда алгоритмов (например, поиск кратчайшего пути) и т.п

Язык программирования Паскаль (Pascal) создан швейцарцем Н.Виртом (Niclaus Wirth) на кафедре информатики Стэнфордского университета на базе структурного программирования. Первая публикация описания языка относится к 1970 г. Особенность языка - его четкая структурированность, ясная логика определений, лаконичность описаний. Популярность язык приобрел после создания Андерсом Хейльсбергом (Anders Hejlsberg) в компании Borland версии Турбо Паскаль, в которой компилятор был объединен с редактором текста. Хейльсберг руководил разработкой среды Delphi, в которой Pascal стал объектно-ориентированным языком Object Pascal.

Язык Си (C) создан в 1972 г. Д.Ритчем (D.Ritchie) и К.Томпсоном (K.Thompson) из Bell Labs/Lucent Technologies при поддержке Б.Кернигана. Ими же в 1973 г. на языке Си

представлена операционная система UNIX. В рекламных целях UNIX в университетах распространялся почти бесплатно, вместе с UNIX распространялся и Си. Благодаря эффективности исполнения программ, написанных на этом языке, он получил широкое распространение. До сих пор Си – один из лучших языков для системного программирования. Он стандартизован в 1989 г., стандартная версия языка обозначается ANSI C.

Первый полностью объектно-ориентированный язык программирования SmallTalk создан в 1972 г Он стал самым значительным практическим результатом работы Алана Кея в компании Xerox.

В 1973 г. в Марсельском университете разработан язык логического программирования Пролог, в котором описываются факты и отношения между ними.

Универсальный язык программирования Ада был разработан по заказу министерства обороны США в 1979 г.. Список требований к языку прошёл через несколько этапов утверждения. Руководителем группы разработчиков был француз Жан Ихбиа.

Вернувшись из Стенфорда в Швейцарию, Н.Вирт развивает методологию программирования. В 1980 г. в языке Модула-2 он реализует концепцию модульного программирования, затем в Оберон-2 – объектно-оринетированное и в Component Pascal – компонентно-ориентированное программирование.

Создание объектно-ориентированного языка C++ относят к 1984 г. Его автором является датчанин Бьерн Страуструп (Bjarne Stroustup), сотрудник лаборатории AT&T Bell Labs, в которой уже были разработаны операционная система UNIX и язык программирования Си, взятый за основу (вместе с идеями объектной ориентированности из языка Симула67) и для C++. Авторы не предполагали, что язык получит широкое распространение, они создавали его для себя, чем возможно и объясняется некоторая нестрогость его описания (как и языка Си).

Язык программирования Java разработан в 1995 г. Джеймсом Гослингом из компании Sun Microsystems.

Ведущую роль в мире в создании программного обеспечения для персональных компьютеров играет компания Microsoft, созданная в 1975 г. Биллом Гейтсом (Gates) и Полом Алленом. В 1980 г. появилась операционная система MS-DOS (Microsoft Disk Operation System) для первого IBM PC. Затем Гейтс разрабатывает программы электронных таблиц Excel и текстовый редактор Word. В дальнейшем Microsoft регулярно обновляет основные компоненты программной системы MS Office, включая ОС Windows.

В 2000 г. Андерс Хейльсберг, перешедший из Borland в Microsoft, разработал язык C# (Си-шарп), легший в основу среды .NET Framework компании Microsoft, и продолжает заниматься развитием этой среды.

В силу определенных причин работы отечественных программистов не получили столь же широкого распространения, как американских и западноевропейских. Однако в СССР и РФ проводились интересные разработки языков, программных систем и сред. В этом отношении можно сослаться на технологию создания крупномасштабных программных средств для систем реального времени, созданную по проекту "Прометей", язык обработки символьной информации Рефал, разработанный В.Ф.Турчиным в МГУ, язык моделирования дискретных и дискретно-непрерывных систем НЕДИС, язык Аналитик для

машин «Мир» и др. Руководителем проекта "Прометей" был В.В.Липаев, являвшийся председателем координационного совета Министерства радиопромышленности СССР по автоматизации проектирования программного обеспечения и написавший ряд книг по технологиям разработки, надежности и качеству программных систем. В Новосибирске А.П.Ершов разрабатывает методы создания программирующих программ и компиляторов.

Интеллектуальные системы.

История интеллектуальных систем тесно соприкасается с историей искусственного интеллекта (ИИ).

Термин искусственный интеллект (artificial intelligence) предложен в 1956 г. в Стэнфордском университете (США). Но история искусственного интеллекта имеет далекую предысторию.

Впервые идею создания искусственного разума, подобного человеческому мозгу, высказал в XIV веке Р.Луллий (ок.1235-ок.1315). В XVIII в. Г.Лейбниц (1646 - 1716) и Р.Декарт (1596 - 1650) независимо друг от друга продолжили попытки создания искусственного интеллекта на основе классификации понятий всех наук.

В 40-х гг. XX века после создания ЭВМ, в частности, благодаря работам Н.Винера (1894 - 1964), идеи создания интеллектуальных систем встали на реальную почву.

Работы в области искусственного интеллекта можно классифицировать по типам решаемых задач, выделяя задачи распознавания образов, доказательства теорем, принятия решений (в том числе в игровых ситуациях), понимание и синтез речи и текстов на естественных языках (ЕЯ). Для решения этих задач на компьютерах необходима разработка соответствующих методов и средств представления и обработки знаний. В этом аспекте искусственного интеллекта различают дедуктивные системы, математическую лингвистику и языки ИИ, нечеткие множества, экспертные системы, многоагентные системы, нейрокибернетику. Применяемые методы опираются на одну из двух парадигм.

Одна из них основана на стремлении использовать аналогии с явлениями живой природы. Во-первых, это направление изучения функционирования человеческого мозга и поиска путей его имитации в ИИ. Так, нейрокибернетика (или нейроинформатика) ориентирована на аппаратное моделирование структур, подобных структурам мозга. Во-вторых, изучение природы наследственности и использование эволюционных принципов в технических системах. Близко к ИИ примыкает бионика и применение ее принципов в робототехнических системах.

Вторая парадигма не связана со структурой мозга и природными реализациями живых организмов. Это направление "черного ящика". т.е. устройства, которое должно выполнять сложные функции, которые считались лишь уделом мыслящих существ, однако "черный ящик" никак не связывался с устройством мозга. Это направление искусственного интеллекта ориентировано на поиски алгоритмов решения интеллектуальных задач на существующих моделях компьютеров.

Понятие исчисления (дедуктивной системы) ввел в 1943 г. американский математик Э.Пост (E.Post). Подход к автоматическому доказательству теорем описал Эрбран (J.Herbrand) в 1930 г., но реализация процедуры доказательства стала возможной после

появления ЭВМ, особенно после разработки метода резолюций Д.Робинсоном (J.Robinson) в 1965 г.

Метод резолюций нашел свое эффективное применение в логическом программировании и языке Пролог, созданном в Марселе Ковальским и Колмероэ в 1973 г. Их первоначальная программа, написанная на Фортране, предназначалась для построения систем обработки текстов на естественных языках. Эффективность Пролога была продемонстрирована после разработки компилятора Пролога в конце 70-х.

Но первым языком искусственного интеллекта, не потерявшим своей популярности до наших дней, был созданный в 1958 г. Джоном Мак-Карти (J.McCarthy) из Массачусетсткого технологического института язык обработки списков ЛИСП (LISP - LISt Processing), ставший языком функционального программирования.

В реальных задачах детерминированность является лишь модельным приближением, не всегда адекватным исследуемой ситуации. Это относится и к математической логике. Для отражения неопределенности, присутствующей в практических задачах вследствие неполноты исходных данных, Л.Заде (Lotfi Zadeh) к 1965 г. разработал теорию нечетких множеств, нашедшую широкое применение в системах ИИ, в частности, в экспертных системах.

Экспертные системы, основанные на выявлении и представлении в компьютере человеческих знаний, привлекли к себе заметное внимание в середине 70-х годов. Одними из первых были разработаны ставшие уже классическими экспертные системы MYCIN и DENDRAL для медицины и химии.

MICIN — экспертная система для медицинской диагностики. Разработана группой по инфекционным заболеваниям Стэнфордского университета. Ставит соответствующий диагноз, исходя из представленных ей симптомов, и рекомендует курс медикаментозного лечения любой из диагностированных инфекций. База данных состоит из 450 правил.

DENDRAL — экспертная система для распознавания химических структур. Первые версии данной системы появились еще в 1965 году также в Стэнфордском университете. Пользователь дает системе DENDRAL некоторую информацию о веществе и данные спектрометрии, а система выдает диагноз в виде соответствующей химической структуры.

К числу первых из разработанных экспертных систем относится также PROSPECTOR — экспертная система, созданная для содействия поиску коммерчески оправданных месторождений полезных ископаемых.

В 1969 г. в Электротехнической лаборатории (Япония) началась разработка проекта "промышленный интеллектуальный робот". Цель этой разработки — создание очувствленного манипуляционного робота с элементами искусственного интеллекта для выполнения сборочно-монтажных работ с визуальным контролем

Трудно назвать конкретную дату, которую можно было бы назвать днем рождения многоагентных систем (МАС), поскольку многие события в истории ИИ так или иначе были прелюдией к созданию МАС [18]. Среди этих событий нельзя не отметить работы У.Р.Эшби (Ashby), М.Л.Цетлина, Д.А.Поспелова и др. Заметный вклад в теорию МАС внес К.Хьюитт (C.Hewitt), рассматривавший в 1977 г. распределенную систему, как совокупность взаимодействующих акторов [19]. Использование МАС направлено на снижение сложности систем управления, повышение их живучести, росту

быстродействия, так как процессы легче распараллеливаются, происходит локализация большинства связей на малых расстояниях.

Генетические алгоритмы (ГА) – наиболее значительное представление эволюционных вычислений. Д.Холланд (J.Holland) [20] признан в мире как основоположник ГА. Он и его последователи Д Голдберг (D.Goldberg), Де Янг (De Jong) и др. разработали основы и ряд стратегий генетического поиска при решении задач оптимизации и принятия решений в различных приложениях.

В нейроинформатике первые исследования искусственных нейронов и нейронных сетей были выполнены П.Мак-Каллоком и Питсом в 1943 г. [21].

В 1958 г. Фрэнк Розенблатт разработал однослойную нейронную сеть, названную перцептроном, и построил первый нейрокомпьютер «Марк-1». [22]. Это была попытка создать систему, моделирующую человеческий глаз и его взаимодействие с мозгом. Перцептрон умел различать буквы алфавита, но был чувствителен к их написанию.

М.Минский [23] показал, что с помощью однослойного перцептрона можно решать лишь существенно ограниченный круг практических задач. Заметных успехов в 70-80 гг. добиться не удалось, что привело к снижению интереса к перцептронам и в значительной мере к свертыванию работ по реализации идей искусственного интеллекта.

По мере роста производительности и объема памяти ЭВМ восстанавливался интерес к интеллектуальным системам. Важным стимулом возобновления работ в середине 80-х гг. было создание в Японии проекта компьютера V поколения, основанного на знаниях. Появились транспьютеры — параллельные компьютеры с большим количеством процессоров. От транспьютеров был один шаг до нейрокомпьютеров, моделирующих структуру мозга человека.

Для практического применения нейросетей необходима разработка алгоритмов их обучения. Первые алгоритмы обучения многослойных нейронных сетей были предложены в СССР А.И.Галушкиным в 1973 г.

В 1986 г. Дэвидом Румельхартом с соавторами опубликованы основы метода обучения многослойного перцептрона, названного ими «методом обратного распространения ошибки» (error back-propagation) для нейронных сетей, хотя основные положения этого метода уже были представлены в магистерской диссертации П.Вербоса (P.Werbos) в 1974 г.

Сеть встречного распространения разработана Р.Хехт-Нильсеном в 1987 г. [24].

Сеть Хопфилда [25] за счет введения обратной связи устанавливается в одно из возможных своих состояний. Каждому состоянию соответствуют определенный экстремум функции **Y**(**X**), где **Y** - выходной, а **X** - входной векторы сети, и область притяжения к экстремуму в пространстве векторов **X**. Если сеть обучена, то подача на вход некоторого **X**, находящегося в области притяжения экстремума **X'**, приводит к переходу сети в устойчивое состояние **Y**(**X'**). Тем самым решаются задачи распознавания, ассоциативной памяти, кластеризации и т.п.

Первыми нейрокомпьютерами были перцептроны Ф.Розенблатта Марк-1 (1958 г.) и Тобермори (1964 г.). К числу первых нейрокомпьютеров относится также Адалин, разработанный Б.Уидроу [26], и некоторые другие. Во второй половине 80-х годов

появляются нейрокомпьютеры на микропроцессорах, примерами которых могут служить ANZA и DELTA, в 90-е годы создаются нейрокомпьютеры на сигнальных процессорах.

Достаточно весомы результаты работ по проблемам искусственного интеллекта, проводившихся в России.

В 1954 г. в МГУ под руководством профессора А.А.Ляпунова (1911 - 1973) начал свою работу семинар "Автоматы и мышление". В этом семинаре принимали участие крупнейшие физиологи, лингвисты, психологи, математики.

В 1959 г. по инициативе А.И.Берга создается Научный совет по комплексной проблеме "Кибернетика" при Президиуме АН СССР. Аксель Иванович Берг (1893-1979) создал в АН СССР Институт радиотехники и электроники, способствовал созданию Института семиотики, стоял у истоков такого научного направления как вычислительная лингвистика (вместе с А.А.Ляпуновым).

В 1955 - 1964 гг. создаются отдельные программы и исследуется поиск решения логических задач. В Ленинграде (ЛОМИ — Ленинградское отделение математического института им. В.А.Стеклова) создается программа, автоматически доказывающая теоремы (АЛПЕВ ЛОМИ). Она основана на оригинальном обратном выводе С.Ю.Маслова, аналогичном методу резолюций Робинсона.

В 1965-1980 гг. получает развитие новая наука— ситуационное управление (соответствует представлению знаний в западной терминологии). Основоположник этой научной школы — профессор Д.А.Поспелов. Разработаны специальные модели представления ситуаций — представления знаний.

В Московском государственном университете создается язык РЕФАЛ.

Развитию генетических алгоритмов в СССР и России способствовали работы Л.А.Растригина, Ю.И.Неймарка, И.Л.Букатовой и др.

Первой советской системой по моделированию автономных агентов стала ТАИР, разработанная под руководством Н.М.Амосова.

Одной из сфер приложений искусственного интеллекта, позволяющей сравнивать возможности естественного и искусственного интеллектов является игра в шахматы. Над программами шахматной игры трудятся целые коллективы. В 1974 г. состоялся турнир шахматных программ, который выиграла советская программа Каисса.

В 90-е годы разработаны отечественные нейрокомпьютеры серии "Геркулес" и компьютеры на нейрочипах транспьютерного типа, в разработке которых принимали участие А.И.Галушкин, Ю.П.Иванов и др.

Автоматизация и методология проектирования. Методология проектирования сложных систем основана на принципах и подходах системотехники.

Необходимость проектирования сложных систем стала важным стимулом для развития системотехники. как части общей теории систем, возникшей в середине XX века, благодаря трудам Л.фон Берталанфи, М.Месаровича, Дж Клира и др Идеи системотехники нашли свое выражение в блочно-иерархическом подходе к проектированию ЭВМ [28-29] и в обобщении этого подхода применительно к проектированию в различных областях техники [30].

Были сформированы основные положения методологии проектирования:

1) Разделение процесса проектирования на иерархические уровни и аспекты с возможными итерационными циклами.

2) Унификация элементной базы. В автомобиле-, самолето-, судостроении - везде, где создаются сложные системы, - используют типовые узлы. Времена, когда вычислительная машина проектировалась "с нуля" и включала разработку как блоков и устройств, так и ламповых ячеек, ушли в далекое прошлое. Теперь проектирование ЭВМ и ее элементной базы разделены. Цифровая аппаратура проектируется на унифицированных ПЛИС и IP-блоках, заказные СБИС - на стандартных ячейках, программные системы - на основе библиотек классов..

3) Использование нисходящего проектирования (с элементами восходящего стиля при применении типовых компонентов).

4) Автономность архитектурных компонентов сложной системы, ведущее к распределенности управления. Этот подход нашел свое воплощение в многоагентных системах.

5) Разумное ограничение сложности систем с целью упрощения их использования. Примеры такого ограничения можно видеть в создании RISC-процессоров или в применении специализированных устройств вместо универсальных.

6) Бездефектность проектирования, что особенно важно для интегральных схем и бортовой аппаратуры. Для выполнения этого положения важно иметь тестовые наборы с высокой полнотой проверки проектов, использовать технологии проектирования контролепригодной аппаратуры, в том числе технологии самотестирования.

Наиболее трудным для автоматизации является концептуальное проектирование, когда отсутствуют формальные модели, а требования к проектируемой системе конкретизированы в недостаточной мере. Для автоматизации концептуального проектирования предназначены средства CASE (Computer Aided System Engineering). К числу наиболее известных CASE-средств относятся методики IDEF, методология объектно-ориентированного проектирования, включая язык UML, и поддерживающее их программное обеспечение.

Ряд методик концептуального проектирования IDEF (Integated DEFinition) создан в США по программе Integrated Computer Aided Manufacturing. Среди них имеются методики функционального, информационного и поведенческого моделирования и проектирования. Методика функционального моделирования IDEF0 создана на основе разработанной Р.Россом в 1973 г. методики SADT (Structured Analysis and Design Technique). Информационное моделирование поддерживает методика IDEF1X, поведенческое моделирование – IDEF3. Эти методики позволяют структурировать представления о проектируемой системе и выразить их в удобной для восприятия графической форме.

Объектно-ориентированное проектирование применяется при разработке преимущественно информационных систем со сложным программным обеспечением. В программной индустрии языки объектно-ориентированного программирования (ООП) выражают концепцию разработки программных систем, противопоставляемую процедурному программированию. Собственно идеи ООП нашли применение в ряде алгоритмических языков, начиная с Симула-67 и Smalltalk. Однако использование ООП не как языка программирования, а как технологии проектирования программных систем

определено в методике, разработанной в 1996 г. Гради Бучем (Booch), Айваром Джекобсоном и Джеймсом Рамбо. Они создали UML (Unified Modeling Language) — язык для спецификации, виртуализации, конструирования и документирования информационных систем.

Язык UML был официально принят OMG в качестве стандарта в 1997 г. Поддержкой языка UML занимается компания Rational Software, специализирующаяся на разработке CASE-средств, систем автоматизированного проектирования ПО, средств управления программными проектами.

Методики IDEF и язык UML вошли в арсенал CALS-средств.

Автоматизация проектирования вычислительных машин.

Автоматизация проектирования зародилась в радиоэлектронной промышленности, значительно опередив по времени появление САПР в области машиностроения. Очевидной причиной этого является необходимость в машиностроительных САПР (MCAD) выполнять основной объем работ конструкторского характера, что требует использования высококачественных графических станций, но такие станции стали доступными лишь в 80-е годы.

Что касается САПР цифровых автоматов и вычислительных машин, первые результаты для создания таких САПР были получены в конце 50-х годов. Эти результаты в значительной мере были предвосхищены трудами английского математика XIX века Джорджа Буля (1815-1864), заложившего основы математической логики; создателя математической теории информации Клода Шеннона, занимавшегося вопросами теории релейно-контактных схем; советского физика В.И.Шестакова, который одновременно с К.Шенноном (1938 г.) предложил применять математическую логику к синтезу логических схем.

Разработка теории логического анализа и синтеза релейно-контактных, а позже и функциональных схем на основе аппарата математической логики в СССР в 40-х годах была продолжена М.А. Гавриловым и далее С.В. Яблонским , В.М.Глушковым , Д.А.Поспеловым и др.

М. А. Гаврилов создал стройную теорию анализа и синтеза одно- и многотактных релейно-контактных схем, являющуюся составной частью прикладной теории автоматов и дискретных устройств. Его первая работа по теории релейно-контактных схем относится к 1943 г. Работы М.А.Гаврилова по блочному синтезу явились толчком к развитию целой серии работ по методам композиции и декомпозиции автоматов. В области САПР М. А. Гавриловым была создана диалоговая автоматизированная система логического проектирования дискретных устройств и систем.

Дмитрий Александрович Поспелов окончил МГУ в 1956 г. Он автор ряда методов построения систем управления, в основе которых лежит идея семиотических (логико-лингвистических) моделей представления объекта управления и описания процедур управления ими. Дмитрием Александровичем был создан аппарат ярусно-параллельных форм, позволивший ставить и решать многие проблемы, связанные с организацией параллельных вычислений в вычислительных комплексах и сетях. Заметную роль в развитии автоматизации логического проектирования сыграла его книга "Логические методы анализа и синтеза схем".

Работы по автоматизации проектирования вычислительных машин были направлены, во-первых, на проектирование функциональных и логических схем, во-вторых, на конструирование печатных плат и оформление конструкторской документации.

Логическое моделирование применяется с конца 50-х годов. Одной из первых программ логического моделирования была программа, о которой сообщили С.Крей и Р.Киш в 1956 г. в США . В 1965 г. американские разработчики САПР начинают использовать трехзначное логическое моделирование. В 1966 г. Дж. Рот разрабатывает свой знаменитый *d*-алгоритм синтеза тестов для контроля и диагностики неисправностей в схемах . В 70-е годы разрабатывается концепция автоматизации проектирования цифровых систем с выделением уровней системного, регистровых передач, логического и схемотехнического.

История САПР вычислительных машин в нашей стране тесно связана с такими организациями как ИТМиВТ, НИИ автоматической аппаратуры, НИИ молекулярной электроники, ЦКБ Алмаз, ведущими инженерными вузами. В СССР еще в середине 1957 г. В.М.Глушков определяет направления стратегических исследований в области информатики, отнеся к ним разработку методов автоматизации проектирования ЭВМ и развитие методов автоматизации программирования. В 1964 г. И.Я.Ландау предложил язык моделирования логических схем ФОРОС . В 1965 г. Г.Г.Рябов в ИТМиВТ начал разработку САПР, позднее получившую название ПУЛЬС, а Н.Я.Матюхин возглавил работы по автоматизации проектирования ЭВМ. В 1967 г. вопросами САПР ЭВМ начинает заниматься О.Н.Юрин, который в 70-е годы возглавляет разработку САПР ЕСАП (Единая Система Автоматизации Проектирования) в НИЦЭВТе. В Киеве принципиальные вопросы автоматизации проектирования вычислительных машин разрабатывает В.М.Глушков с коллегами .

Значительное внимание уделяется автоматизации конструкторского проектирования печатных плат и интегральных схем. Алгоритмы построения минимальных покрывающих деревьев были предложены в 50-е годы Д.Краскалом и Р.Примом, несколько позднее опубликован волновой алгоритм трассировки С.Ли . При размещении предлагаются последовательные и итерационные алгоритмы, в частности, алгоритм парных перестановок, появившийся в 1960 г.

В СССР методы и программы конструкторского проектирования РЭА (радиоэлектронной аппаратуры) и БИС во второй половине 60-х годов разрабатывают Г.Г.Рябов, Л.Б.Абрайтис, В.А.Селютин и др.

В начале 70-х годов работы по созданию САПР получили признание. В 1972 г. Государственной премией СССР было отмечено создание САПР для радиотехнической промышленности (позднее эта система стала называться ПРАМ), разработчиками которой были Е.И.Бронин, Ю.Х.Вермишев, Л.П.Рябов и др. Годом позже Государственная премия СССР присуждена специалистам из НИИ молекулярной электроники во главе с Г.Г.Казенновым за разработку САПР для микроэлектронной промышленности. В эти САПР входили программы компоновки, размещения и трассировки печатных плат (ПРАМ) или кристаллов БИС (НИИМЭ), моделирования электронных и логических схем, синтеза тестов и др.

Автоматизация проектирования на функционально-логическом и системном уровнях во-многом связана с созданием языков проектирования (design language).

На функционально-логическом уровне проектирования радиоэлектронной аппаратуры и СБИС наиболее известны языки VHDL, Verilog, SystemC. При конструкторском

проектировании для описания топологии СБИС широкую известность получил формат EDIF (Electronic Design Interchange Format).

Язык VHDL (Very high-speed integrated circuits Hardware Design Language) - язык моделирования дискретных электронных устройств, утвержденный в качестве международного стандарта IEEE 1076 в 1987 г.. В дальнейшем стандарт корректировался и расширялся, новые версии приняты в 1993 и 1999 г.г., в частности, в версии IEEE 1076.1 (1999 г.) нашли отражение особенности описания аналоговых устройств. Язык предназначен для моделирования преимущественно на уровнях вентильном, регистровых передач и корпусов микросхем, он успешно используется и при синтезе устройств.

К числу предшественников VHDL можно отнести один из первых языков для описания схем на уровне регистровых передач APL (1962 г.), разработанный в США.

Для моделирования на системном уровне было разработано большое число языков. Из числа общецелевых языков моделирования одним из первых был язык GPSS [45], появившийся в 1964 г. и, что удивительно, продолжающий широко использоваться и в настоящее время. В 60-е годы разработаны еще несколько известных языков системного моделирования. К ним относятся Simscript [46], Симула-67 [47] и ряд других. Наряду с моделями массового обслуживания, при системном моделировании используют аппарат, разработанный Петри (C. A. Petri) в 1962 г. и названный сетями Петри.

САПР в электронике.

История САПР в электронике берет свое начало в первой половине 60-х годов прошлого века.

В США первыми программами анализа нелинейных электронных схем были TAP, NET-1 , разработанные в 1962 и 1964 г. соответственно, а также более известная программа ECAP. У истоков автоматизации проектирования в электронике стояли Ф.Брэнин, Д.Калахан, Р.Рорер и др. В 80-е годы проблемами автоматизации проектирования СБИС на логическом и схемотехническом уровнях активно занимаются А.Ньютон, А.Санджованни-Винчентелли, С.Дайректор и др.

Первая в СССР программа анализа электронных схем разработана в МВТУ им. Н.Э.Баумана И.П.Норенковым, сообщение о ней появилось в 1965 г. Это была программа ПАЭС для ЭВМ Урал-2. В ней были использованы более ранние работы по моделированию полупроводниковых приборов, выполненные Д.Эберсом и Д.Моллом в США, С.Е.Жорно в СССР. В 70-е годы были созданы программы аналогичного назначения в Зеленограде В.П.Панферовым, в МАИ - В.Н.Ильиным, в МИФИ - А.Я.Архангельским, в Киеве - В.П.Сигорским и А.И.Петренко и др.

Американские ЭВМ были более быстродействующими, но советские программы не уступали американским по времени решения задач за счет использования разреженности матриц в математических моделях схем.

Уже в конце 60-х годов стала ясной необходимость комплексного подхода к автоматизации проектирования, обеспечивающего сквозной цикл разработки как БИС и СБИС, так и печатных плат. Создание САПР БИС и СБИС в министерстве электронной промышленности СССР возглавляет НИИ молекулярной электроники. Работы ведутся под руководством сначала Г.Г.Казеннова, затем главного конструктора САПР МЭП Б.В.Баталова.

Разработка средств моделирования электронных схем стимулировала развитие численных методов решения возникающих задач. С 1972 г. разработчики программ анализа перешли на использование неявных методов интегрирования систем дифференциальных уравнений. Появляются А-устойчивые методы (Гира, неявно-явные), обобщаются методы разреженных матриц, разрабатываются методы ускоренного моделирования быстроосциллирующих процессов.

С ростом степени интеграции микроэлектронных схем задачи проектирования становятся все более сложными. Разработка БИС и СБИС без автоматизации проектирования уже невозможна. Для преодоления сложностей топологического проектирования СБИС С.Мид и Д.Конвей в 1980 г. разрабатывают систему автоматического проектирования топологии, названную кремниевым компилятором [51] и основанную на применении совокупности правил преобразования высокоуровневого описания схемы в реальную топологию.

Появляются компании, целиком специализирующиеся на создании средств ECAD. Среди них выделяются три гранда - Mentor Graphics, Cadence, Synopsys.

Современный период микросхемных ЭВМ

1978 г. — Выпущен процессор 8086, первый 16-разрядный микропроцессор Intel; впервые применена очередь команд; появилась возможность подключать математический сопроцессор (8087); 4,77МГц (позже появились МП 8 и 10 МГц), 330 тыс. оп./с, технология 3мкм, 29000 транзисторов, адресация 1MB ОЗУ. Выпущен процессор 8088 (Intel), 16-разрядный микропроцессор с 8-разрядной шиной данных, ставший «сердцем» первого IBM PC. Atari представляет персональные компьютеры Atari 400 и Atari 800, работающие на микропроцессоре 6502 компании MOS Technology. Язык программирования Modula-2 Никлауса Вирта, Microsoft выпускает свой 3-й язык программирования — Microsoft Cobol-80.

1979 г. — Выпущен процессор MC68000 (Motorola), 16-разрядный, адресация 16MB ОЗУ.

1980 г. — Вышло три процессора (Motorola 68000, Intel 80186, NS 16000) и появилась первая настольная СУБД (Dbase II). Создан первый портативный компьютер Osborne 1 весом около 12 кг. Первый компьютер IBM PC, открывший эру IBM-совместимых ПК, созданный в рамках проекта Chess; CPU — Intel 8088, RAM — 16KB, HDD — 5,25" 160KB, ОС — MS-DOS 1.0 или CP/M.

1981 г. — На рынок выходит IBM PC (Intel 8088 4,77 МГц, 16 Кбайт RAM, FDD 160 Кбайт) с цветным монитором. Можно было приобрести примерно за 1600 долларов. С подачи компании Sony появляются трехдюймовые дискеты. Появляется первый успешно продаваемый переносной микрокомпьютер с экраном, дисководами и сумкой для переноса (прообраз ноутбуков) Osborne 1, разработанный корпорацией Osborne Computer. MS-DOS 1.0 компании Microsoft, PC-DOS 1.0 анонсировала IBM.

1982 г. — Появление первой версии AutoCAD и языка PostScript. Стараниями Sony появляются звуковые компакт-диски. Винт Серф и Боб Кан создают черновой вариант TCP/IP. Примерно в это же время появляется термин Internet. Выпущен процессор 80286 (Intel), 134 тыс. транзисторов, технология 1,5мкм, адресация до 16MB ОЗУ; принципиальное новшество — защищенный режим. Выпущен видеоадаптер HGC с разрешением 720x348x2 (Hercules) и CGA 640x200 пикселей.

1983 г. — IBM выпустила персональный компьютер XT (сокращение от eXtended Technology). Официальным днем рождения считается 8 марта.

1984 г. — Sony и Philips разрабатывают стандарт CD-ROM. Также разработаны стандарты MIDI и DNS. В продаже появляются IBM PC AT, более мощные варианты персоналок, работающие на новом чипе от Intel — 80286. Apple выпускает модем на 1200 бод. Hewlett-Packard выпускает первый лазерный принтер серии LaserJet с разрешением до 300 dpi.

1985 г. — Видеоадаптер EGA с разрешением 640x350x16 (IBM). Выходит первая версия графической оболочки Windows, поддерживающей многозадачность, суперкомпьютер производительностью 1 млрд. операций в секунду (Cray 2) и новый язык программирования C++. В СССР выпущен бытовой компьютер «Электроника БК0010-01» и начат выпуск многопроцессорного (10 процессоров) вычислительного комплекса Эльбрус-2 производительностью 125 млн. оп/сек (MIPS). В.С.Бурцев, который использовался в системах противоракетной обороны, ядерных центрах, центре контроля космического пространства. Выпущен процессор Intel 386, 32-разрядный многозадачный процессор, содержащий 275 тыс. транзисторов, технология 1,5мкм, адресация до 4GB. Выпущены оптические диски с однократной записью (CD-ROM). 20 ноября — Microsoft Windows 1.0

1986 г. — Появляются первые экспериментальные 4- и 16-мегабайтные чипы памяти. На клавиатуре впервые появляются клавиши управления курсором (до этого обходились без них!) и отдельный блок с цифровыми клавишами (спасибо Apple). В том же году Ларри Уоллом разработан язык Perl (Practical Extraction and Report Language), универсальный язык программирования, применяющийся для составления сценариев CGI (Common Gateway Interface). Mips Technologies представила 32-разрядный RISC-процессор, работающий с тактовой частотой 8МГц, содержащий 110 тыс. транзисторов, с производительностью 5 млн. операций в секунду. В СССР начался выпуск одной из самых популярных машин линии СМ — 16-разрядная микро-ЭВМ СМ 1810, построенная на процессоре с системой команд Intel 8086, и которая уже могла выступать в роли персонального компьютера. СМ 1810 создавалась в Институте электронных управляющих машин (ИНЭУМ). Компания Gateway 2000 выпустила свой первый ПК. ПК Deskpro 386 вывел компанию Compaq Computer в лидеры рынка ПК на база новых процессоров Intel 80386. Американский национальный институт стандартов (ANSI) одобрил стандарт SCSI-1 (Small Computer System Interface). Рождение шины USB. Вышло постановление правительства СССР о начале разработки вычислительного комплекса «Эльбрус-90 Микро».

1987 г. — U.S. Robotics представляет модем Courier HST 9600. Первый релиз операционной системы OS/2, разрабатываемой совместно IBM и Microsoft, и первый релиз IBM PS/2. Весна — Объявление IBM о OS/2 1.0.

1988 г. — Появляется первый компьютерный вирус, известный нам как «червь Морриса», созданный сыном эксперта по компьютерной безопасности. Изобретен IRC-чат. Компании, занимавшиеся клонированием IBM PC, столкнувшись с необходимостью лицензирования новой шины IBM — MCA, разрабатывают собственную — EISA. Выпущен видеоадаптер VGA с разрешением 640x480x16 (IBM) и SVGA — 800x600. Выпущены магнито-оптические (MO) и фазопеременные диски.

1989 г. — Тим Бернерс-Ли разрабатывает концепцию WWW. Creative Labs выпускают звуковую карту Sound Blaster, название которой впоследствии станет нарицательным. Выпущен процессор Intel 486DX, 1,2 млн транзисторов, технология 1мкм, первичный кэш на кристалле, применено RISC-ядро. Первая версия программы Adobe Photoshop.

1990 г. — В СССР выпущена модификация бытового компьютера БК0011М. IBM представляет новый стандарт видеоплат — XGA. Появляется Интернет. В Институте точной механики и вычислительной техники им. С.А.Лебедева разработан микропроцессор «Электроника Эль-90», предназначенный для разрабатываемого с 1986 года вычислительного комплекса «Эльбрус-90 Микро». Процессор так и не был произведен в связи с политическими и экономическими преобразованиями в стране.

1991 г. — Создан опытный образец ЭВМ «ЛОКОН 9В51» (проект «ЛОКОН» — локально-связанная машина). ЭВМ построена в архитектуре CLIP/CAM (клеточно-автоматные машины), в основу проекта были положены принципы параллельной обработки информации с расширяемой архитектурой. Главный конструктор и научный руководитель проекта — Бронников В. А.

В СССР создан Эльбрус-3 — LSI, ECL БИС, 16 процессоров, быстродействие в два раза выше, чем у лучшего западного суперкомпьютера CRAY-YMP, был изготовлен, но в серию запущен не был (на конечной стадии в работе участвовало дочернее предприятие Института — Московский центр SPARC-технологий).

1993 г. — В марте выпущен процессор Pentium (Intel) с частотами 60МГц (индекс ICOMP — 510) и 66МГц (ICOMP — 567), 32-разрядный процессор с 64-разрядной шиной данных, 3,1 млн транзисторов, технология 0,8 мкм, рабочее напряжение — 5В. В октябре выпущен процессор Intel Pentium с частотой 75 МГц (индекс ICOMP — 610); технология 0,6 мкм, рабочее напряжение 3,3 В. 4 известные компании — Compaq Computer, Intel Corporation, Microsoft и Phoenix Technologies предложили решение проблемы конфигурации IBM PC-совместимых компьютеров, разработав спецификацию Plug and Play, которая определяла средства и способы взаимодействия периферийных устройств с BIOS компьютера и ОС, а также могла разрешать конфликты из-за системных ресурсов с минимальным участием пользователя.

В России (Москва) начался выпуск персональных компьютеров PS/1.

1994 г. — В марте выпущен процессор Intel Pentium с частотами 90МГц (ICOMP — 735) и 100МГц (ICOMP — 815) процессоров Pentium, технология 0,6мкм, рабочее напряжение — 3,3В.

1995 г. — Фирма Phoenix Technologies создала первый Plug and Play BIOS, который был использован в новых системах NEC и Gateway 2000. Выпущен процессор Intel Pentium Pro (P6) с частотой 150МГц, 5,5 млн транзисторов, технология 0,6 мкм, вторичный кэш на кристалле 256KB. Pentium с частотами 120МГц (ICOMP — 1000) и 133МГц (ICOMP — 1110), технология 0,35 мкм, рабочее напряжение — 3,3 В. Cyrix 6x86. Intel Pentium Pro с частотами 166, 180 и 200 МГц, 0,35мкм технология; 512KB кэш 2-го уровня. Появился Iomega Zip — накопитель на сменных дискетах 100MB.

1996 г. — Выпущены процессоры Intel Pentium с частотами 150, 166 и 200 МГц. В марте выпущен AMD K5. Разработана технология перезаписываемых CD (CD-RW). Корпорация Sun Microsystems представила 64-разрядное семейство рабочих станций Ultra.

1997 г. — В январе выпущен Intel Pentium MMX, 4,5 млн транзисторов, 0,35мкм технология; выпущены МП с частотами 166, 200 и 233 МГц. В феврале выпущен Cyrix Media GX. Также выпущен Intel Pentium II, использующий технологию MMX, 7,5 млн транзисторов; кристалл с ядром процессора и набор кристаллов статической памяти и дополнительных схем, реализующих вторичный кэш, размещены на небольшой печатной

плате-картридже; выпущены МП с частотами 233, 266 и 300 МГц. Первые дисководы DVD. Выпуск первых звуковых плат формата PCI. Новый графический порт AGP.

В России в ЗАО «МЦСТ» продолжена разработка вычислительного комплекса «Эльбрус-90 Микро», начатая в институте точной механики и вычислительной техники им. С.А.Лебедева еще в 1986 году.

1998 г. — В России в ЗАО «МЦСТ» создан SPARC-совместимый микропроцессор с технологическими нормами 0,5 мкм и частотой 80 МГц (из неофициальных источников процессор назывался МЦСТ-R100 и работал на частоте 50 МГц). На его основе в том же году создан прототип первого Российского микропроцессорного вычислительного комплекса «Эльбрус-90 Микро».

В марте Advanced Micro Devices выпустила K6. Появился LS-120 (Super Disk) — накопитель на дискетах 120MB.

1999 г. — В феврале выпущен Intel Pentium III.

2000 г. — 20 ноября выпущен Intel Pentium IV, на основе микроархитектуры NetBurst, 42 млн транзисторов, микросхема 217кв.мм в 423-контактном корпусе PPGA, тактовые частоты 1,4 и 1,5ГГц (позже появился 1,3ГГц), технология 0,18мкм. В июне компания IBM создала новый суперкомпьютер серии RS/6000 SP — ASCI White (Accelerated Strategic Computing Initiative White Partnership) — первый компьютер, производительность которого превышает 10 TFLOPS. Пиковая производительность суперкомпьютера — 12,3 TFLOPS; компьютер способен постоянно работать на скорости 3 TFLOPS. ASCI White представляет собой 512 компьютеров, соединенных вместе, и по площади занимает 2 баскетбольные площадки; а для его транспортировки понадобится 28 трейлеров. Компьютер разработан для Национальной лаборатории Лоуренса Ливермора Министерства энергетики США, где он будет использоваться для моделирования ядерных взрывов и контроля за хранящимся американским ядерным оружием.

2000–2005 гг. — Вариации на тему Pentium III и Pentium IV (2000 — до 2 ГГц, 2001 — до 2,8 ГГц, 2002 — до 3,06 ГГц, 2003 — до 3,2 ГГц, 2004 — до 3,8 Ггц, 2005 — двухядерные процессоры с теми же частотами).

2001 г. — В России ЗАО «МЦСТ» разработан SPARC-совместимый микропроцессор «МЦСТ R150» с технологическими нормами 0,35 мкм и тактовой частотой 150 МГц.

У Intel появились процессоры Xeon для рабочих станций и Itanium, в основе которого лежит новая архитектура EPIC (Explicitly Parallel Instruction Computing — параллельная обработка команд с явным параллелизмом).

2002 г. — Первый Российский микропроцессорный вычислительный комплекс «Эльбрус-90 Микро» на базе отечественного микропроцессора «МЦСТ-R». Разработан ЗАО «МЦСТ».

Новый Celeron выполнен на основе ядра Willamette по 0.18 мкм процессу. Отличается от Pentium IV на том же ядре вдвое меньшим объёмом кэша второго уровня (128 против 256 Kb).

2003 г. — В России ЗАО «МЦСТ» разработан SPARC-совместимый микропроцессор «МЦСТ R500» с технологическими нормами 0,13 мкм и тактовой

частотой 500 МГц для использования в вычислительном комплексе «Эльбрус-90 Микро». Рассеиваемая мощность процессора менее 1 Вт.

2004 г. — В России ЗАО «МЦСТ» разработан SPARC-совместимый микропроцессор на полностью заказной технологии с топологическими нормами 0,13 мкм и тактовой частотой 1000 МГц а также микропроцессор нового поколения «Эльбрус» с топологическими нормами 0,13 мкм и тактовой частотой 300 МГц.

На основе процессора «МЦСТ R500» в 2004–2005 годах построено пять модификаций вычислительного комплекса «Эльбрус-90 Микро».

2005 г. — В апреле Intel выпущен первый двухядерный процессор для настольных ПК — Pentium Extreme Edition.

В России ЗАО «МЦСТ» разработан микропроцессор «E2K» нового поколения на полностью заказной технологии с топологическими нормами 0,13 мкм и тактовой частотой 1200 МГц (он же «Эльбрус-2000). Этот микропроцессор построен по не имеющей аналогов передовой отечественной технологии, в которой реализована архитектура явного параллелизма (EPIC), уже использовавшаяся в 2001 году в процессоре Intel Itanium. В связи с дороговизной производства процессор «E2K» скорее всего выпущен не был, хотя информация об этом неоднозначна. Разговоры о завершении разработки этого процессора идут уже несколько лет. Создаётся впечатление, что разработчик специально запутывает общественность — путаница существует как на его сайте http://www.mcst.ru так и в других источниках. Существует мнение, что «E2K» — афёра.

Конец 2005 г. — В России ЗАО «МЦСТ» в конце декабря передана на фабрику документация для изготовления опытных образцов микропроцессора «Э3М» второй итерации — упрощенный вариант «сказочного» процессора «E2K». микропроцессор «Э3М» работает на частоте 300 МГц и не имеет (в отличие от «E2K») кэша второго уровня.

Также были проведены конструкторские испытания опытного образца вычислительного комплекса «Эльбрус-3М1», основанного на процессоре «Э3М». По результатам испытаний принято решение о передаче откорректированной топологической документации на фабрику. Передача документации успешно выполнена 30 декабря 2005 года.

2006 г. — В январе Intel выпущен новый двухядерный процессор «Core» для настольных и мобильных ПК. Выпущена также одноядерная версия этого процессора для мобильных ПК. Название Core теперь заместило слово Pentium. Вероятно, эти процессоры можно считать новым 10-ым поколением (80086, 80186, 80286, 80386, 80486, Pentium, Pentium I, Pentium II, Pentium III, Pentium IV, Core). Вообще, поколения процессоров у Intel, начиная с процессора Pentium, а в особенности с Pentium IV, стали весьма условными. Современный процессор Intel включает в себя несколько технологий, применяемых вместе, которые постепенно друг за другом обновляются, и четкой границы между поколениями уже провести нельзя.

4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Лабораторные работы имеют различный уровень сложности и на их выполнение требуется различное количество часов. Каждая предполагает самостоятельную работу студентов по освоению лекций и теоретического материала, вынесенного на самостоятельное изучение.

ЛАБОРАТОРНАЯ РАБОТА № 1. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА JAVA, ПРИЛОЖЕНИЯ JAVA.

Двумя основными формами Java-программ являются приложение и апплет. Далее рассматриваются различия между этими приложениями и их назначение на основе создания простейшего кода как для приложения, так и для апплета.

Java-программы могут выполняться под управлением специального интерпретатора (java.exe), работающего в рамках отдельного процесса, либо под управлением навигатора Интернет, такого, как Microsoft Internet Explorer или Netscape Navigator. В последнем случае программа называется апплетом.

Java-приложение работают независимо от навигатора, главное их отличие от апплетов лежит в их назначении. Приложения похожи на программы, созданные, например, с использованием языка C/C++, хотя для своей работы они требуют присутствия среды Java. Но, в отличие от апплетов, их существование никак не связано с Internet и они не выполняются как содержимое страниц WWW. Это полноправные приложения, которые существуют и выполняются в локальных компьютерных системах пользователей.

Java-апплеты же разработаны для функционирования в сети и выполняются как часть станиц WWW, поэтому к ним относятся как к исполняемому содержимому. Хотя они и встраиваются в страницы WWW подобно стандартному содержимому, созданному с использованием HTML, на самом деле это программы, которые запускаются и выполняются.

Апплеты требуют наличия соответствующего Java-броузера, так как они должны загружаться по сети с сервера WWW в обеспечивающую их работоспособность среду исполнения Java на локальном компьютере.

1. Простейшее приложение Hello

Перед созданием приложения познакомимся с последовательностью действий для создания приложений:

Использование JDK (Java Developer's Kit).

1. Создание, ввод и сохранение обычного тестового файла, содержащего код программы, имеющего расширение .java (например, Hello.java). Использовать можно любой текстовый редактор, позволяющий работать с файлами, имеющими длинные имена, например Notepad.
2. Компиляция исходного кода Java в машинный байтовый код при помощи компилятора javac. В результате трансляции создаются файлы с расширением .class (Hello.class).
3. Исполнение приложения: передача файла байтового кода интерпретатору java для выполнения приложения.

Использование среды разработки JBuilder.

1. Создание нового Java-проекта с именем Hello (меню "File", пункт "New Project")
2. В диалоговом окне "Project Wizard" задаем имя проекта Hello.java в поле "Name". Нажимаем кнопку "Finish". В области "Project" появится дерево с именем проекта Hello.java.jpx .
3. Для добавления файла в созданный проект в контекстном меню выбираем "Add Files/Packages". В диалоговом окне "Add Files or Packeges to Project" в закладке "Explorer" задаем имя файла Hello.java в поле "File name". Файл Hello.java должен быть включен в проект Hello.

4. Двойной щелчок мыши по имени файла Hello.java в области “Project” открывает рабочую область файла, где в закладке “Sourse” вводится текст программы.

5. Для запуска приложения в области “Project” из контекстного меню файла Hello.java выбираем пункт “Run”.

2. Структура Java-программы

Все Java-программы содержат в себе четыре разновидности блоков: классы (classes), методы (methods), переменные (variables) и пакеты (package).

Методы есть не что иное как функции или подпрограммы. В *переменных* же хранятся данные. Данные понятия присутствуют так или иначе во всех языках программирования. С другой стороны, *классы* представляют собой фундамент объектно-ориентированных свойств языка. Для простоты на данном этапе изучения языка Java можно сказать, что класс - это некое целое, содержащее переменные и методы.

Наконец, *пакеты* содержат в себе классы и помогают компилятору найти те классы, которые нужны ему для компиляции пользовательской программы. Классы, входящие в один пакет, особым образом зависят друг от друга, пока же, опять-таки для простоты, можно рассматривать просто как наборы классов. Например, приложение Hello импортирует пакет java.util, в котором содержится класс Date.

Java-программа может содержать в себе любое количество классов, но один из них всегда имеет особый статус и непосредственно взаимодействует с оболочкой времени выполнения (*первичный класс*). В таком классе обязательно должны быть определены один (для приложений) или несколько (для апплетов) специальных методов. Для приложений первичный класс должен обязательно содержать метод main().

2.1 Переменные

Переменную можно представить как хранилище для единицы данных, имеющее собственное имя. Любая переменная принадлежит определенному *типу*. Тип переменной определяет, какую информацию в ней можно хранить. Переменные должны быть объявлены с использованием следующего синтаксиса (модификаторы рассматриваются ниже в разделе, посвященном классам):

<Модификаторы> ТипПеременной ИмяПеременной ;

В Java существует два вида переменных. Первый - *примитивные типы* (primitive types). К ним относятся стандартные, *встроенные в язык* типы для представления численных значений, одиночных символов и булевских (двоичных, логических) значений. Все примитивные типы имеют *предопределенный размер* занимаемой ими памяти. Ко второму виду переменных - *ссылочные типы* (reference type) - относятся типы, определенные пользователем (классы, интерфейсы) и типы массивов. Все ссылочные типы являются *динамическими* типами, для них выделяется память во время работы программы.

Примитивные и ссылочные типы также различаются по тому, как переменные этих типов передаются в качестве параметров методам (то есть функциям). Переменные примитивных типов передаются *по значению*, тогда как ссылочные переменные всегда передаются *по ссылке*.

Практически самым важным различием между двумя типами переменных является то, что память для ссылочных переменных выделяется динамически, во время выполнения программы. Использование переменных ссылочных типов требует явного запрашивания требуемого количества памяти для каждой переменной прежде, чем можно будет сохранить в этой переменной какое-либо значение. Причина проста - оболочка времени выполнения сама по себе не знает, какое количество памяти требуется для того или иного ссылочного типа.

Переменные-массивы и другие ссылочные переменные лишь указывают на то место в памяти, где содержатся собственно данные, тогда как переменные примитивных типов ни на что не указывают, а просто содержат в себе соответствующие данные, имеющие определенный фиксированный размер.

Ссылочные переменные, хоть и очень похожи на указатели C/C++, имеют сильное отличие от них. Используя ссылочные типы, нельзя получить доступ к фактическим адресам данным в памяти.

2.1.1 Примитивные типы

Всего в Java определено восемь примитивных типов: int (4b), short (2b), byte (1b), long (8b), float (4b), double (8b), boolean (true, false, 1 бит?), char (2b).

Первые шесть типов предназначены для хранения численных значений, с ними можно производить арифметические операции. Тип char предназначен для хранения символов в стандарте Unicode.

Булевские (логические, двоичные) переменные могут иметь одно из двух допустимых значений: true или false. Двоичные константы являются именно константами, а не строками. Их нельзя преобразовать в строковый тип. Они также не являются целыми значениями нуль или единица, как двоичные значения в языках C/C++.

Булевским переменным можно присваивать не только булевские константы, но и результаты сравнения переменных различных типов. Операции !, !=, == работают с булевскими значениями так же, как одноименные операторы работают с целочисленными значениями в языке C/C++.

2.1.2 Ссылочные типы

Ссылочные типы отличаются от примитивных тем, что они не определены в самом языке Java, и поэтому количество памяти, требуемое переменных этих типов, заранее знать невозможно. Пример одного из ссылочных типов - это тип массива. Массивы языка Java могут состоять из переменных любых типов, включая типы, определенные пользователем

Язык Java не позволяет просто объявить переменную ссылочного типа и сразу же начать записывать в нее значение. Необходимо сначала запросить у оболочки времени выполнения некоторый объем памяти, а оболочка, в свою очередь, должна сделать запись в своих внутренних таблицах, что активизирована переменная данного ссылочного типа. Весь этот процесс в целом и называется *реализацией* переменной. После реализации, когда уже имеется экземпляр переменной данного типа, можно использовать этот экземпляр для хранения данных. Важно понимать, что экземпляр переменной и сам ссылочный тип, к которому эта переменная относится, являются качественно различными понятиями - для хранения переменной можно использовать только реализованный экземпляр переменной ссылочного типа.

Типы массива

Типы массива используются для определения массивов - упорядоченного набора однотипных переменных. Можно определить массив над любым существующим типом, включая типы, определенные пользователем. Кроме того, можно пользоваться и массивами массивов или многомерными массивами.

2.2 Методы

Метод представляет собой подпрограмму, аналогичную функциям языка C и Pascal. Каждый метод имеет тип возвращаемого значения и может вызываться с передачей некоторых параметров.

С каждым методом должен быть соотнесен тип возвращаемого им значения. Например, тип void является специальным способом указать системе, что данный метод не возвращает никакого значения. Методы, у которых возвращаемое значение принадлежит к любому другому типу, кроме void, должны содержать в своем теле оператор return. Возвращаемое значение может принадлежать к любому из типов, включая примитивные типы и ссылочные типы.

В качестве параметров в языке Java можно передавать переменные любого типа, включая типы, определенные через классы, и массивы переменных любого типа и размера. Однако в качестве параметров переменные примитивных типов ведут себя иначе, чем переменные ссылочных типов.

Все переменные примитивных типов передаются методам по значению (by value). Это означает, что в момент вызова метода делается копия переменной, передаваемой методу. Если метод в своем теле будет изменять значение переданной ему в качестве параметра переменной, то содержимое исходной переменной изменяться не будет, так как все действия будут производится с ее копией.

Напротив, значения переменных ссылочного типа, переданных в качестве параметров, можно изменить в теле метода. Когда методу в качестве параметра передается переменная ссылочного типа, то при изменении ее значения явным образом меняется содержимое того, на что указывает эта переменная.

Иногда возникает необходимость создавать две или несколько функций, выполняющих по сути, одни и те же действия, но имеющие различные списки параметров. В языке Java можно присвоит одно и то же имя нескольким методам, которые различаются списками своих параметров. Этот процесс называется *совмещением методов*.

Выгоды совмещения методов особенно очевидны в том, вместо запоминания нескольких имен разных методов можно ограничиться запоминанием одного только имени, общего для всех методов с разными параметрами. В обязанности компилятора входит выяснение того, какой именно метод требуется вызвать в каждом случае.

2.3 Классы

В Java переменные и методы класса могут быть либо *элементами класса*, либо *элементами экземпляра* класса, в котором они объявлены, что определяется присутствием или отсутствием модификатора static**.**

Если при определении элемента не используется ключевое слово static, то это элемент по умолчанию является *динамическим* (dynamic). Динамические методы и переменные всегда являются *элементами экземпляра класса*, и доступ к ним осуществляется через переменную-объект.

Статические методы и переменные связаны с классом, а не с экземпляром класса, и поэтому имеют название *элементов класса*. Элементы класса можно использовать без создании объекта этого класса, доступ осуществляется через имя класса. Каждая *переменная класса* и каждый *метод класса* уникальны *в своем классе*; *методы* и *переменные экземпляра* уникальны в своем экземпляре класса. Различие между членами класса и экземпляра значительно, в особенности, если дело касается переменных

Элементы класса, будучи уникальными в своем классе, используются всеми объектами, созданными из этого класса, - то есть все объекты, созданные из данного класса, разделяют статические переменные и методы, определенные в этом классе (для всех объектов существует только один экземпляр статической переменной). *Элементы экземпляра класса*, с другой стороны, создаются каждый раз, когда создается объект.

Элементы класса могут, таким образом, считаться глобальными относительно класса, несмотря на то, что настоящие глобальные переменные в Java не поддерживаются. Когда какой-нибудь объект класса изменяет значение *переменной класса*, результат становится видим всем объектам. Благодаря этому переменные класса часто используют в качестве общих данных для всех объектов, созданных из этого класса.

2.3.2 Модификаторы доступа

Модификаторы доступа используются для управления доступностью элементов класса из других частей программы (в других классах). Применение ключевых слов public (открытый), protected (защищенный), private protected и private (закрытый) к элементам помогает управлять способностью других объектов пользоваться ими.

Элемент, объявленный с ключевым словом *public*, доступен во всех классах как в том пакете (о пакетах классов см. ниже), в котором он был объявлен, так и во всех классах в любом другом пакете. Из всех модификаторов данный накладывает наименьшие ограничения на доступность элемента- он доступен всем, является открытым для всех. Этот модификатор, в отличие от всех остальных, можно использовать и *при объявлении класса* (класс может иметь этот модификатор или не иметь). Тогда этот класс также

доступен для всех других классов невзирая на то, частью какого пакета классов он является. В каждом файле должен содержаться только один открытый класс.

Элемент, объявленный с модификатором *protected* в некоем классе A, доступен во всех классах в данном пакете, а также во всех классах, являющихся подклассами класса A. Иными словами, доступа к этому элементу нет в тех классах, которые не входят в данный пакет и не являются подклассами того класса, в котором этот элемент определена.

Если же элемент в классе A объявлен как *private protected*, то это означает, что к нему можно получить доступ только в подклассах класса A. В других же классах, даже входящих в этот же пакет, этот элемент недоступен.

Модификатор private сильнее всего ограничивает доступность элемента. Он его делает невидимым нигде за пределами данного класса. Даже подклассы данного класса не смогут обращаться к элементу, объявленному как private.

Модификатор доступа (private, protected и private protected) используется для так называемого скрытия данных (data hiding). Объявление переменных или методов класса с закрывающим их модификатором делает невозможным их использование вне области доступности. Методы же их собственного класса имеют к ним доступ всегда.

2.3.3 Наследование классов

Модификаторы доступа делают классы более устойчивыми и надежными в работе, так как гарантируют, что снаружи класса можно будет получить доступ только к некоторым из методов и переменных. *Наследование* (inheritance), в свою очередь, упрощает практическое использование классов, так как позволяет *расширять* уже написанные и отлаженные классы, *добавляя* к ним новые свойства и возможности.

Если класс явно не объявляется подклассом какого-либо класса, компилятор предполагает по умолчанию его подклассом класса Object.

Задания к лабораторной работе

Задание 1. Создать следующие приложения *Hello* , *VarTypes*, *NewClass* , *TestMethods* **,** *TestModificators* .

ЛАБОРАТОРНАЯ РАБОТА № 2. СОЗДАНИЕ ПРОСТЕЙШИХ АППЛЕТОВ.

Апплет встраивается в документ HTML (HTML - язык для разметки гипертекста на WWW-страницах) и выглядит как окно заранее заданного размера. Он может рисовать в своем окне (и только в нем) произвольные изображения или текст.

Двоичный файл с исполняемым (а точнее говоря, интерпретируемым) кодом Java располагается на сервере WWW. В документ HTML с помощью специального оператора организуется ссылка на этот двоичный файл. Когда пользователь загружает в навигатор документ HTML с апплетом, файл апплета переписывается с сервера WWW на локальный компьютер пользователя. После этого навигатор начинает его выполнение.

Использование JDK (Java Developer's Kit).

1. Создание, ввод и сохранение обычного тестового файла, содержащего код программы, имеющего расширение .java (например, Hello.java). Использовать можно любой текстовый редактор, позволяющий работать с файлами, имеющими длинные имена, например Notepad.

2. Создание с помощью того же текстового редактора файла HTML (например, Hello.html), в который встраивается созданный апплет. Для этого в него включается следующий тег:<APPLET CODE=Hello.class WIDTH=200 HEIGHT=200></APPLET>

3. Компиляция исходного кода Java в машинный байтовый код при помощи компилятора javac. В результате трансляции создаются файлы с расширением .class (Hello.class).

4. Исполнение приложения:

1) Просмотр файла Hello.html с помощью WWW-навигатора, поддерживающего работу апплетов.

2) Либо использование программы просмотра апплетов appletviewer.exe.

Замечание. Для выполнения компиляции и запуска приложения можно создать

командный файл (с расширением .bat) следующего содержания:

javac.exe Hello.java

appletviewer.exe Hello.html

6. Создание нового Java-проекта с именем Hello (меню “File”, пункт “New Project”).

7. В диалоговом окне “Project Wizard” задаем имя проекта Hello.java в поле “Name”. Нажимаем кнопку “Finish”. В области “Project” появится дерево с именем проекта Hello.java.jpx .

8. Для добавления файла в созданный проект в контекстном меню выбираем “Add Files/Packages”. В диалоговом окне “Add Files or Packeges to Project” в закладке “Explorer” задаем имя файла Hello.java в поле “File name”. Файл Hello.java должен быть включен в проект Hello.

9. Двойной щелчок мыши по имени файла Hello.java в области “Project” открывает рабочую область файла, где в закладке “Sourse” вводится текст программы.

10. Для добавления HTML-файла в созданный проект в контекстном меню выбираем “Add Files/Packages”. В диалоговом окне “Add Files or Packeges to Project” в закладке “Explorer” задаем имя файла Hello.html в поле “File name”. Файл Hello.html должен быть включен в проект Hello.

11. Двойной щелчок мыши по имени файла Hello.html в области “Project” открывает рабочую область файла, где в закладке “Sourse” вводится текст HTML-файла.

12. Компиляция исходных текстов.

13. Для запуска приложения в области “Project” из контекстного меню файла Hello.html выбираем пункт “Run”.

Апплет Hello, управляемый мышью

Класс Applet содержит большое количество методов, которые вызываются в ответ на действия пользователя (например, перемещение курсора мыши в пределах окна или нажатие определеннх клавиш на клавиатуре). Приведем в качестве примера использование метода mouseDown(), который вызывается каждый раз, когда пользователь в пределах области, занятой апплетом, нажимает левую клавишу мыши. Усовершенствованная программа Hello перерисовывает строчку текста в той точке, где пользователь щелкнул мышью.

Простейший апплет HelloApplet, созданный Java Applet Wizard

Для создания шаблона апплета, на основе которого можно разрабатывать специализированные апплеты, можно воспользоваться системой автоматизированной разработки шаблонов апплета Java Applet Wizard, встроенной в JBuilder.

Создание шаблона апплета HelloApplet

Для создания шаблона апплета HelloApplet среде разработки JBuilder выбрать пункт “New” меню “File”. В появившейся диалоговой панели “Object Gallery” выбрать закладку “Web” и отметить тип “Applet”. В диалоговом окне “Applet Wizard” в поле “Class” следует ввести имя HelloApplet и нажать кнопку “Finish”.

В результате работы системы Java Applet Wizard будет создано два файла с исходными текстами: текст апплета (файл HelloApplet.java) и HTML-документ (файл HelloApplet.html).

Исходные файлы апплета HelloApplet

Исходный текст апплета начинается с двух строк, в которых с помощью оператора import подключаются библиотеки классов.

Далее в исходном тексте апплета определяется класс типа public с именем HelloApplet, которое должно совпадать с именем файла, содержащего исходный текст этого класса.

Метод paint() выполняет рисование в окне апплета. Определение этого метода находится в классе java.awt.Component. Так как класс Applet является подклассом класса Component (см. иерархию классов апплетов), а класса HelloApplet - подклассом Applet, то метод paint() можно переопределить в классе HelloApplet.

Метод paint() вызывается, когда необходимо перерисовать окно апплета. Перерисовка окна апплета обычно выполняется асинхронно по отношению в работе апплета (подобно перерисовки клиентской части окон Windows-приложений при поступлении сообщения WM_PAINT). В любое время апплет должен быть готов перерисовать содержимое своего окна.

Методу paint() в качестве параметра передается ссылка на объект класса Graphics. По своему смыслу этот объект напоминает контекст отображения, который используется для вывода информации в Windows-окно. Контекст отображения - это как лист бумаги, на котором можно рисовать изображение или выводит текст. Многочисленные методы класса Graphics позволяют задавать различные параметры вывода, такие, например, как цвет или шрифт.

Для вывода изображения апплеты используют координатную систему, которая соответствует режиму MM_TEXT - одному из режимов, использующихся при программирование для Windows. Начало этой системы координат расположено в левом верхнем углу окна аплета, ось X направлена слева направо, а ось Y - сверху вниз.

Аргументы апплета

Аргументы командной строки передаются апплетам во время запуска и происходит при помощи специально созданных атрибутов апплетов. Эти *атрибуты параметров апплетов* (или проще - параметры апплетов) определяются в HTML-теге <APPLET> и предоставляют соответствующую информацию же самому апплету.

Параметры апплетов следуют после открывающего тега <APPLET> и перед закрывающим тегом </APPLET>. Они определяются как пары, состоящие из двух опций - NAME (имя) и VALUE (значение), внутри тегов <PARAM>.

Для того, чтобы апплет принимал параметры, при создании апплета с помощью системы Java Applet Wizard в четвертой диалоговой панели необходимо определить параметры, передаваемые апплету.

Первоначально список параметров пуст, для того чтобы добавить новый параметр, нужно в столбце “Name” ввести имя параметра. Для апплета AppletWithParam следует добавить параметры с именами String_1 и String_2.

В столбце “Member” при заполнении списка параметров отображаются имена полей класса AppletWithParam, в которые можно будет записать значения параметров (при помощи метода getParameter()). Значения столбца “Def-Value” используются для инициализации соответствующих полей класса. Следует задать в этих полях строки “First string” и “Second string” соответственно.

Для описания параметров служит столбец “Description”. Апплет может извлечь эту информацию методом getParameterInfo() и проанализировать.

Загрузить изображение можно при помощи URL-адреса на файл с изображением. URL, или Uniform Resource Locators (“унифицированная ссылка на ресурс”), - полный адрес объекта в сети WWW.

В Java есть отдельный класс для обработки URL-адресов - java.net.URL. Самый простой способ создания объекта URL состоит в использовании конструктора URL(String add), передавая ему обычный WWW-адрес, например

URL addUrl=new URL("http://www.vvsu.ru/");

Нужно отметить, что фрагмент кода, где происходит реализация класса URL, может сгенерировать исключение MalformedURLException, если вдруг объект не может по какой-либо причине быть создан (например, если передаваемая конструктору строка не является URL-адресом). Компилятор требует обработать возможность возникновения исключительной ситуации при помощи блоков try-catch.

Класс URL содержит конструктор, которому передается абсолютный базовый URL и строка, содержащая путь к объекту относительно этого базового указателя. Создать URL на файлы своего сервера (или своего локального компьютера) можно при помощи именно этого конструктора. Для получения URL на каталог, содержащий класс работающего апплета, используется метод getCodeBase() класса Applet. Так имя загружаемого

графического файла содержится в переменной m_FileName класса, то для загрузки графического изображения можно использовать следующий фрагмент, помещаемый в метод init() класса ParamUrlImage:

Для ускорения вывода картинок на экран и для устранения эффекта мерцания изображений в процессе вывода большинство апплетов использует *двойную буферизацию изображения* - сначала загружая изображение в оперативную память, а затем выводя его в окне апплета за один шаг.

Каждый раз, когда апплет вызывает метод drawImage(), он создает поток, вызывающий метод imageUpdate(), который можно переопределить в классе апплета и использовать для того, чтобы определить, какая часть изображения загружена в память. Поток, созданный метод drawImage(), вызывает метод imageUpdate() до тех пор, пока он возвращает true.

Параметр infoflags этого метода позволяет отследить, какая часть изображения загружена в память (есть несколько определенных состояний этого флага). Когда этот параметр равен ALLBITS, это означает, что это изображение полностью находится в памяти.

События и их обработка

Многие обычно используемые обработчики событий определены для класса Applet и по умолчанию не делают ничего - чтобы их использовать, надо переопределить метод суперкласса апплетов в классе создаваемого апплета.

Метод mouseMove() вызывается всякий раз при перемещении мыши. После обработки события метод возвращает true.

Перечислим часто используемые обработчики событий:

- boolean mouseDown(Event evt, int x, int y) - нажата кнопка мыши, параметры x и y указывают расположение мыши.
- boolean mouseUp(Event evt, int x, int y) - кнопка мыши отпущена.
- boolean mouseMove(Event evt, int x, int y) - перемещение мыши.
- boolean mouseDrag(Event evt, int x, int y) - перемещение мыши с нажатой кнопкой.
- boolean mouseEnter(Event evt, int x, int y) - перемещение мыши на окно апплета.
- boolean mouseExit(Event evt, int x, int y) - мышь покинула окно апплета.
- boolean keyDown(Event evt, int key) - нажата клавиша перемещения курсора или функциональная клавиша. Параметр key указывает эту клавишу.
- boolean keyUp(Event evt, int key) - клавиша перемещения курсора или функциональная клавиша отпущена.

Переопределяя эти методы в подклассе класса Applet, можно обрабатывать различные действия пользователя.

Задания к лабораторной работе

Задание 1. Должны быть созданы следующие апплеты: *Hello* , **модифицированный апплет** *Hello* , *AppletWithParam* , *ParamUrlImage* , *QuickPicture*, *MouseEvent*

ЛАБОРАТОРНАЯ РАБОТА № 3. РИСОВАНИЕ В ОКНЕ, ОБРАБОТКА СОБЫТИЙ МЫШИ И КЛАВИАТУРЫ .

Рисование в окне

В процессе рисования на экране прежде всего требуется наличие объекта класса Graphics (контекст отображения). Графические операции всегда выполняются над объектом Graphics. Например, в апплетах для вывода в окно используется метод paint(), которому передается единственный параметр - объект класса Graphics.

Некоторые методы класса Graphics

- Graphics - Конструктор нового объекта - контекста отображения
- clearRect - Очищает указанный прямоугольник, заполняя цветом фона

- clipRect - Задает область ограничения вывода
- getClipRect - Возвращает ограничивающий прямоугольник области отсечения
- copyArea - Копирует область экрана
- create - Создает новый объект, который является копией исходного объекта
- draw3DRect - Рисует прямоугольник с объемным эффектом
- drawArc - Рисует дугу текущим цветом
- drawBytes - Рисует указанные байты текущим шрифтом и цветом
- drawChars - Рисует указанные символы текущим шрифтом и цветом
- drawImage - Рисует указанное изображение типа Image
- drawLine - Рисует линию между точками
- drawOval - Рисует овал внутри указанного прямоугольника текущим цветом
- drawPolygon - Рисует многоугольник текущим цветом
- drawRect - Рисует контур прямоугольника текущим цветом
- drawRoundRect - Рисует контур прямоугольника с закругленными краями
- DrawString - Рисует указанную строку текущим шрифтом и текущим цветом
- fill3DRect - Раскрашивает цветом прямоугольник с объемным эффектом
- fillArc - Заполняет дугу текущим цветом
- fillOval - Заполняет овал текущим цветом
- fillPolygon - Заполняет многоугольник текущим цветом
- fillPolygon - Заполняет объект класса Polygon текущим цветом
- fillRect - Заполняет прямоугольник текущим цветом
- fillRoundRect - Заполняет прямоугольник с закругленными краями
- GetColor - Получить текущий цвет
- getFont - Получить текущий шрифт
- getFontMetrics - Получить размеры шрифта
- setColor - Устанавливает текущий цвет
- setFont - Устанавливает текущий шрифт
- setPaintMode - Устанавливает режим заполнения текущим цветом
- setXORMode - Устанавливает режим заполнения
- translate - Сдвиг начала системы координат в контексте отображения

Цвет

Для задания текущего цвета используется метод setColor() класса Graphics:

Цветовая модель языка Java представляет собой 24-разрядную модель RGB (красный, синий, зеленый), следовательно объекты класса Color могут содержать 24 разряда цветовой информации (что соответствует 16 миллионам различных цветов), определяемой в виде составляющих красного, зеленого и синего цветов.

Для использования цвета необходимо сначала создать объект Color и передать в него значения красного, зеленого и синего цвета (причем существуют два конструктора - для задания целочисленных значений (каждое значение от 0 до 255) и значений с плавающей точкой (каждое значение от 0.0 до 1.0)). Совместно эти значения и определяют цвет.

```
Color clr1=new Color(255,0,0); // создать красный цвет
Color clr2=new Color(0,255,0); // создать зеленый цвет
Color clr3=new Color(0,0,255); // создать красный цвет
Color clr4=new Color(255,255,255); // создать белый цвет
Color clr5=new Color(0,0,0); // создать черный цвет
Color clr6=new Color(100, 100, 100); // создать оттенок серого цвета
```

Класс Color содержит ряд методов, которые позволяют осуществить запрос для нахождения различных составляющих цвета (методы getRed(), getGreen(), getBlue(), getRGB()), получения разновидностей текущего цвета (например, методы brighter() и

darker() возвращают соответственно более светлый или более темный оттенок текущего цвета), а также другие методы.

Помимо установки текущего цвета отображения текста и графики методом setColor() класса Graphics, можно установить цвет фона и цвет переднего плана методами setBackground() и setForeground() класса Component.

Шрифты

Класс Graphics позволяет размещать на экране текст с использованием установленного шрифта. Для того чтобы воспользоваться шрифтом, необходимо прежде всего создать объект класса Font. Для этого необходимо лишь предоставить название гарнитуры шрифта (тип String), стиль шрифта (тип int) и размер шрифта в пунктах (тип int):

Font f=new Font("Times Roman",Font.BOLD,72);

Для задания стиля шрифта используются константы класса Font: Font.PLAIN, Font.BOLD и Font.ITALIC. Эти стили можно объединять при помощи операции +. После создания шрифта его можно установить для использования при помощи метода setFont() класса Graphics:

2. Обработка событий

Когда возникает событие, управление получает метод **handleEvent()** из класса Component (класс Applet является подклассом класса Component). Прототип метода handleEvent():

public boolean handleEvent(Event evt);

В качестве параметра методу передается объект класса **Event**, который содержит всю информацию о событии. По содержимому элементов класса Event можно определить координаты курсора мыши в момент, когда пользователь нажал клавишу, отличить одинарный щелчок мыши от двойного и т.д.

Перечислим элементы класса Event, которые можно проанализировать:

- Object target - компонент (объект), в котором произошло событие.
- Object arg - произвольный аргумент события, значение которого зависит от типа события.
- int id - тип события, одна из констант, определенная в классе Event.
- int key - идентификатор клавиши.
- int modifiers - состояние маски наложения клавиш.
- long when - время, в которое произошло событие.
- int x - коодината x события.
- int y - коодината y события.
- int clickCount - равна 1 или 2 в зависимости от того был сделан одинарный или двойной щелчок.

Элемент id (тип события) может содержать следующие значения:

- ACTION_EVENT - пользователь хочет, чтобы произошло некоторое событие, например, он нажал на кнопку, изменил состояние переключателя, выбрал элемент из меню выбора (выпадающего списка), выбрал элемент раскрывающегося списка, для поля ввода нажал клавишу «Enter», выбрал пункт из меню окна.
- GOT_FOCUS - компонента (например, окно апплета) получил фокус ввода.
- KEY_ACTION - пользователь нажал функциональную клавишу F1 - F12 или клавишу перемещения курсора.
- KEY_ACTION_RELEASE - пользователь отпустил функциональную клавишу F1 - F12 или клавишу перемещения курсора.
- KEY_PRESS - пользователь нажал обычную клавишу.
- KEY_RELEASE - пользователь отпустил обычную клавишу.
- LIST_DESELECT - отмена выделения элемента в списке.
- LIST_SELECT - выделение элемента в списке.
- LOAD_FILE - загрузка файла (не используется в апплетах).

- LOST_FOCUS - компонента потеряла фокус ввода.
- MOUSE_DOWN - пользователь нажал клавишу мыши.
- MOUSE_DRAG - пользователь нажал клавишу мыши и начал выполнять перемещение курсора мыши.
- MOUSE_ENTER - курсор мыши вошел в область окна апплета.
- MOUSE_EXIT - курсор покинул область окна апплета.
- MOUSE_MOVE - перемещение курсора мыши без нажатой клавиши.
- MOUSE_UP - пользователь отпустил клавишу мыши.
- SAVE_FILE - сохранение в файле (не используется в апплетах).
- SCROLL_ABSOLUTE - пользователь переместил бегунок полосы просмотра в новую позицию.
- SCROLL_LINE_DOWN - сдвиг на одну строку вниз.
- SCROLL_LINE_UP - сдвиг на одну строку вверх.
- SCROLL_PAGE_DOWN - сдвиг на одну страницу вниз.
- SCROLL_PAGE_UP - сдвиг на одну страницу вверх.
- WINDOW_DEICONIFY - пользователь сделал запрос операции восстановления нормального размера окна после его минимизации.
- WINDOW_DESTROY - пользователь собирается удалить окно.
- WINDOW_EXPOSE - окно будет отображено.
- WINDOW_ICONIFY - окно будет минимизировано.
- WINDOW_MOVED - окно будет перемещено.

Если событие связано с клавиатурой, в элементе key может находится одно из следующих значений:

- код обычной клавиши.
- UP, DOWN, HOME, END, LEFT, RIGHT, PGUP, PGDN.
- F1 - F12.

Для состояния маски наложения клавиш модификаторов modifiers могут быть указаны следующие:

- ALT_MASK - была нажата клавиши <Alt>.
- META_MASK - была нажата метаклавиша.
- CTRL_MASK - была нажата клавиша <Ctrl>.
- SHIFT_MASK - была нажата клавиша <Shift>.

Приложение может переопределить метод handleEvent() и обрабатывать события самостоятельно, однако есть более простой путь. Обработчик handleEvent(), который используется по умолчанию, вызывает несколько методов, которые более удобны в использовании при обработки простых событий от мыши и клавиатуры. Можно переопределить именно эти специальные методы для обработки этих событий.

События от клавиатуры

Апплет может обрабатывать события, создаваемые клавиатурой. Он может реагировать на нажатие и отпускание функциональных клавиш F1 - F12, клавиш перемещения курсора, обычных клавиш.

Задания к лабораторной работе

Задание 1. Создать апплеты *FontsList, LinesDraw, KeyCodes* и объяснить их работу. Апплеты должны иметь возможность работать как независимые приложения.

ЛАБОРАТОРНАЯ РАБОТА № 4 КОМПОНЕНТЫ И ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ

Графическое окружение языка Java основано на классе Component. Всякий графический вывод на экран связан (прямо или косвенно) с компонентом. Например, в силу того, что апплеты фактически являются потомками класса Component, имеется возможность рисовать прямо в апплете, не прибегая для этого к специальным методам.

Класс Component содержит очень много методов для обеспечения работы с компонентом GUI.

Некоторые методы класса Component:

- getParent - Получает объект-предок компонента
- isShowing - Проверяет, является ли компонент видимым
- isEnabled - Проверяет, является ли компонент разрешенным
- location - Возвращает текущее положение компонента
- size - Возвращает текущий размер компонента
- bounds - Возвращает текущие границы компонента
- enable - Разрешает компонент
- disable - Запрещает компонент
- show - Отображает компонент
- hide - Скрывает компонент
- getForeground - Получает текущий цвет переднего плана
- setForeground - Устанавливает текущий цвет переднего плана
- getBackground - Получает текущий цвет фона
- setBackground - Устанавливает текущий цвет фона
- getFont - Получает текущий шрифт
- setFont - Устанавливает текущий шрифт
- move - Перемещает компонент в новое положение (в системе координат предка)
- resize - Изменяет ширину и высоту компонента
- reshape - Изменяет форму компонента (положение, высоту и ширину)
- prefferedSize - Возвращает предпочтительный размер компонента
- minimumSize - Возвращает минимальный размер компонента
- getGraphics - Получает графический контекст компонента
- getFontMetrics - Получает типографские параметры текущего шрифта
- paint - Отвечает за рисование компонента
- update - Отвечает за обновление компонента, вызывается методом repaint()
- paintAll - Отвечает за рисование компонента и его подкомпонентов
- repaint - Вызывает перерисовку компонента. Этот метод вызывает метод update()
- print - Отвечает за печать компонента
- imageUpdate - Отвечает за перерисовку компонента при изменении выводимого в компоненте изображения типа Image
- createImage - Создает изображение
- prepareImage - Подготавливает изображение типа Image для визуализации
- checkImage - Возвращает состояние создания экранного представления изображения
- inside - Проверяет, находится ли заданная точка «внутри» компонента
- locate - Возвращает компонент, содержащий заданную точку
- deliverEvent - Доставляет событие компоненту
- postEvent - Пересылает событие компоненту, в результате чего вызывается метод handleEvent()
- handleEvent - Отвечает за обработку события. Если метод возвращает false, то событие передается предку компонента
- mouseDown, mouseDrag, mouseUp, mouseMove, mouseEnter, mouseExit - Отвечают за обработку соответствующих событий мыши
- keyUp, keyDown - Отвечают за обработку соответствующих событий клавиатуры

- action - вызывается в том случае, если в компоненте происходит некоторое действие
- gotFocus - Указывает на то, что компонент получил фокус ввода
- lostFocus - Указывает на то, что компонент потерял фокус ввода
- requestFocus - Запрашивает фокус ввод
- nextFocus - Передает фокус следующему компоненту

Этими методами могут пользоваться все наследуемые от класса Component классы GUI - все *элементы управления* и класс контейнеров. Эти компоненты GUI имеют следующие деревья наследования:

Object → Component → Button
Object → Component → CheckBox
Object → Component → Choice
Object → Component → List
Object → Component → Scrollbar
Object → Component → Label
Object → Component → Canvas
Object → Component → TextComponent
Object → Component → TextComponent → TextField
Object → Component → TextComponent → TextArea
Object → Component → Container

Для того чтобы нарисовать изображение, вывести некоторый текст или разместить элемент пользовательского интерфейса на экране, должен использоваться контейнер класса Container или его подкласса. *Контейнеры* - это объекты, которые содержат компоненты. А *компоненты* - это объекты, которые попадают под класс Component: кнопки, полосы прокрутки, другие элементы управления. Контейнеры сами являются подклассами Component, что подразумевает возможность их вложения или размещения внутри друг друга.

Кнопки (класс Button) - это устройства, которые дают возможность нажимать на них, чтобы проинициализировать некоторое действие.

Для создания кнопок необходимо в методе класса Container (или его подкласса) просто создать экземпляр объекта Button, дать ему имя и добавить его в контейнер:

```
Button myButtom=new Button("Click me!");
add(myButton);
```

или еще проще:

```
add(new Button("Click me!"));
```

Следует обратить внимание на то, что при создании кнопки не передаются координаты X, Y ее положения. Это происходит потому, что элементы управления, как и все устройства, размещаются на экране в соответствии с правилами, определяемыми *менеджерами компоновки* (*размещения*): при этом координаты не используются. Подробно менеджеры размещения будут рассматриваться в следующей главе.

Создать кнопку можно с заданной меткой (строкой), а можно и без нее. Для этого предназначены соответствующие конструкторы **Button()** и **Button(String label)**. Для изменения и получения метки кнопки существуют методы **setLabel()** и **getLabel()** соответственно.

Меню выбора (выпадающие списки) фактически являются всплывающими меню, они дают возможность создавать список элементов выбора, который всплывает на экране в виде меню.

Применяя меню выбора (класс Choice), можно выбирать только один элемент - множественный выбор не разрешается. При необходимости множественного выбора придется воспользоваться раскрывающимся списком (см. следующий пункт).

Полосы прокрутки к спискам, меню выбора и полям редактирования при необходимости добавляются автоматически. Но можно использовать их и как

независимые, отдельные компоненты для обеспечения прокрутки необходимой информации. Создание линейки прокрутки (класс Scrollbar**)** и добавление ее в контейнер осуществляется, например, следующим образом:

```
   Scrollbar
scrlbar=new Scrollbar.HORIZONTAL,25,15,1,100);
   add(scrlbar);
```

Здесь создается горизонтальная линейка прокрутки, ширина которой на экране равна 15. Начальное значение (текущее положение бегунка линейки прокрутки) установлено равным 25, минимальное возможное значение равно 1, а максимальное - 100.

При помощи методов изменять установки линейки прокрутки. Метод setValue() изменяет текущее положение бегунка, а метод setValues() изменяет параметры полосы прокрутки, задаваемые при ее создании. Для установки строчного и страничного инкремента полосы прокрутки используются методы setLineIncrement() и setPageIncrement().

Для получении информации о линейке используются следующие методы. Метод getValue() возвращает текущее положение бегунка, методы getMaximum() и getMinimum() - возвращают максимальное и минимальное значение для полосы прокрутки. Для получения строчного и страничного инкремента полосы прокрутки используются методы getLineIncrement() и getPageIncrement().

Ширину полосы прокрутки на экране (величину ее видимой части) можно узнать методом getVisible(), а ориентация полосы прокрутки определяется методом getOrientation().

Поля редактирования типа TextArea может использоваться как для вывода, так и для ввода и редактирования текста. Поля редактирования этого типа состоят из нескольких строк текста и имеют полосы прокрутки. Напротив, поля редактирования типа TextField состоят из одной строки и не имеют полос прокрутки. Оба этих класса являются наследниками класса TextComponent и, за исключением упомянутых различий, во всем аналогичны друг другу.

Задания к лабораторной работе

Задание 1. Создать апплет *AllElements* и объяснить его работу. Апплет должен иметь возможность работать как независимое приложение.

ЛАБОРАТОРНАЯ РАБОТА № 5. КОНТЕЙНЕРЫ КОМПОНЕНТОВ GUI И МЕНЕДЖЕРЫ РАЗМЕЩЕНИЯ

Контейнеры - это объекты (компоненты), позволяющие помещать в себя другие различные компоненты.

Класс контейнеров Container - подкласс класса Component. Существует два вида основных вида контейнеров: *панели* (класс Panel, подклассом которого является класс Applet) и *окна* (класс Window, подклассами которого являются Frame и Dialog). Контейнеры имеют следующие деревья наследования:

Object → Component → Container → Panel
Object → Component → Container → Panel → Applet
Object → Component → Container → Window
Object → Component → Container → Window → Frame
Object → Component → Container → Window → Dialog
Object → Component → Container → Window → Dialog → FileDialog

В случаях сложного интерфейса контейнеры позволяю объединять элементы управления в смысловые группы и по-разному размещать эти группы элементов относительно друг друга. Так например, в окне апплета можно создать несколько панелей, разделяющих его на части. Отдельные панели могут содержать в себе такие компоненты, как кнопки, переключатели и другие компоненты.

Пространство, занимаемое контейнерами, может быть разделено с использованием одного из менеджеров компоновки (менеджеров размещения). По умолчанию каждый

контейнер имеет ассоциированный с ним менеджер компоновки и предназначены они для визуальной организации элементов интерфейса.

Класс Container имеет методы, при помощи которых происходит управление введенными в него компонентами, установка менеджеров размещения и др.

Некоторые методы класса Container:

- countComponents - Возвращает число содержащихся в контейнере компонент
- getComponent - Возвращает компонент контейнера
- getComponents - Возвращает все компоненты контейнера
- add - Добавляет компонент в контейнер
- remove - Удаляет компонент из контейнера
- removeAll - Удаляет все компоненты из контейнера
- getLayout - Указывает на менеджер компоновки данного контейнера
- setLayout - Устанавливает менеджер компоновки данного контейнера
- layout - Выполняет размещение компонент внутри контейнера
- prefferedSize - Возвращает предпочтительный размер контейнера
- minimumSize - Возвращает минимальный размер контейнера
- paintComponents - Отображает компоненты контейнера
- deliverEvent - Отыскивает нужный компонент и доставляет ему событие
- locate - Возвращает компонент, содержащий заданную точку
- insets - Возвращает вкладки контейнера. Они показывают размер границ контейнера.

Панель (класс Panel) является наиболее общим видом контейнеров в Java. Панель можно использовать как внутри другого контейнера (например, фрейма или апплета), так и непосредственно в окне WWW-броузера**.** Когда интерфейс состоит из большого количества элементов, почти всегда есть смысл объединить группы связанных по смыслу элементов с помощью панелей. Панель может иметь свой собственный менеджер размещения (по умолчанию это FlowLayout), независимый от менеджера размещения контейнера, в который эта панель входит.

Класс Panel имеет подкласс Applet, так что апплеты рассматриваются как расширение класса панелей. Панели очень часто используются в апплетах, имеющий сложный графический интерфейс пользователя.

Окна

Окна (класс Window), как и панели, являются общим классом контейнеров. Но в отличие от панелей окно Java представляет собой окно - объект операционной системы, существующий отдельно от окна WWW-броузера или программы просмотра апплетов.

Непосредственно класс Window никогда не используется, а используются три его подкласса Frame, Dialog и FileDialog. Каждый их этих подклассов содержит все функции исходного класса Window, добавляя к ним несколько специфических свойств (см. описание методов этих подклассов ниже).

Некоторые методы класса Window:

- Window - Конструктор, который создает окно, первоначально не видимое. Окно ведет себя как модальное диалоговое окно, т.е. пока это

окно отображается, оно блокирует ввод в другие окна. В качестве параметра конструктору передается владелец окна - объект класса Frame

- show - Отображает окно. Окно появится на переднем плане, если оно было видимым до этого
- dispose - Удаляет окно. Этот метод необходимо вызвать для освобождения ресурсов, занимаемых окном
- toFront - Переносит рамку на передний план окна
- toBack - Переносит рамку на задний план окна

При создании окна обязательным параметром конструктора Window является объект класса Frame. Этот объект можно создать непосредственно при вызове конструктора окна. Так как окна изначально создаются невидимыми, отобразить их на экране можно, лишь вызвав метод show(). Правда, перед этим иногда полезно придать окну нужные размеры и разместить окно в нужной позиции. Если перемещать окно в видимом состоянии, то это может вызвать неприятное мерцание на экране.

Рамки, фреймы

Фрейм (класс Frame) - это объект, который может существовать без всякой связи с окном WWW-броузера. С помощью класса Frame можно организовать интерфейс независимого апплета. Окно, созданное на базе класса Frame, больше всего похоже на главное окно обычного приложения Windows, и может пользоваться большим количеством методов класса Frame. Оно автоматически может иметь главное меню (так как класс Frame реализует интерфейс MenuContainer), для него можно устанавливать форму курсора и пиктограмму. Внутри такого окна можно рисовать. Так как окно класса Frame произошло от класса Container, то в него можно добавлять различные компоненты и панели, так же как в делается для апплетов и панелей. Нужно отметить, что по умолчанию для окон класса Frame устанавливается режим размещения BorderLayout.

Некоторые методы класса Frame:

- getTitle - Возвращает заголовок
- setTitle - Устанавливает заголовок
- getIconImage - Возвращает пиктограмму
- setIconImage - Устанавливает пиктограмму
- getMenuBar - Возвращает ссылку на объект меню класса MenuBar
- setMenuBar - Устанавливает меню
- remove - Удаляет указанную строку меню
- dispose - Удаляет фрейм. Этот метод необходимо вызвать для освобождения ресурсов, занимаемых фреймом
- isResizable - Проверяет, может ли пользователь изменять размер фрейма
- setResizable - Устанавливает флаг разрешения изменения фрейма
- setCursor - Устанавливает вид курсора с использованием констант класса Frame
- getCursorType - Возвращает тип курсора

Диалоги

Кроме окон и фреймов есть особый подкласс окон, называемый диалоговыми окнами или просто диалоги. В отличие от фреймов или обычных окон диалог предоставляет пользователю окно, в котором нужно выполнить те или иные действия, лишь по завершении которых пользователь сможет продолжить работу с программой.

Диалоговые окна.

Диалоговые окна (класс Dialog) используются в основном для одноразового запроса информации у пользователя или для вывода небольших порций информации на экран. Диалоговые окна во всем подобны фреймам, но имеют два важных отличия: во-

первых, они не являются реализацией интерфейса MenuContainer (а следовательно не реализуют панель меню автоматически); во-вторых, они могут иметь *модальность* - это значит, что можно сконструировать диалоговое окно, которое запретит пользователю обращаться к другим окнам (включая и окно WWW-навигатора) до тех пор, пока пользователь не произведет требуемого действия в этом диалоговом окне.

Некоторые методы класса Dialog:

- getTitle - Возвращает заголовок
- setTitle - Устанавливает заголовок
- isResizable - Проверяет, может ли пользователь изменять размер фрейма
- setResizable - Устанавливает флаг разрешения изменения фрейма
- isModal - Определяет, является ли диалоговая панель модальной

Для окон класса Dialog устанавливается режим размещения BorderLayout. Если нужен другой режим размещения, то необходимо установить его явным образом методом setLayout().

Для отображения окна диалоговой панели необходимо вызвать метод show(). Чтобы спрятать окно, необходимо использовать метод hide(). Метод dispose удаляет окно диалоговой панели окончательно и освобождает все связанные с ним ресурсы.

Диалоговые окна удобны для приглашения пользователя к выполнению какого-либо действия и для подтверждения того, что пользователь ознакомился с сообщением программы. Диалоговые окна в отличие от фреймов не реализуют панель меню автоматически. Поскольку диалоговые окна обычно меньше по размерам, чем фреймы, они бывают удобны для быстрого диалога с пользователем.

При работе с диалоговыми окнами важно помнить одно правило: каждое диалоговое окно обязательно должно иметь фрейм в качестве родителя. Это значит, что диалоговое окно нельзя открыть непосредственно из апплета. Чтобы создать диалоговое окно, необходимо сначала завести фрейм, даже если его единственным значением будет служить родителем для диалогового окна. Только если апплет уже использует фреймы, можно обойтись без этой подготовительной стадии.

Задания к лабораторной работе

Задание 1. Создать апплеты *PanelsDemo1***,** *PanelsDemo2* и *WindowsDemo* и объяснить их работу. Первые два апплета должны иметь возможность работать как независимые приложения.

ЛАБОРАТОРНАЯ РАБОТА № 6.МНОГОПОТОКОВЫЕ ПРИЛОЖЕНИЯ

Процессы, задачи и приоритеты

Обычно в мультизадачной операционной системе выделяют такие объекты, как процессы и задачи. Между этими понятиями существует большая разница, которую следует четко представлять.

Процесс (задача)

Процесс (process) - это объект, который создается операционной системой, когда пользователь запускает приложение. Процессу выделяется отдельное адресное пространство, причем это пространство физически недоступно для других процессов. Процесс может работать с файлами или с каналами связи локальной или глобальной сети.

Поток (нить)

Для каждого процесса операционная система создает одну главный поток (thread), который является потоком выполняющихся по очереди команд центрального процессора. При необходимости главный поток может создавать другие потоки, пользуясь для этого программным интерфейсом операционной системы.

Все потоки, созданные процессом, выполняются в адресном пространстве этого процесса и имеют доступ к ресурсам процесса. Однако поток одного процесса не имеет

никакого доступа к ресурсам потока другого процесса, так как они работают в разных адресных пространствах. При необходимости организации взаимодействия между процессами или потоками, принадлежащим разным процессам, следует пользоваться системными средствами, специально предназначенными для этого.

Приоритеты потоков

Если процесс создал несколько потоков, то все они выполняются параллельно, причем время центрального процессора (или нескольких центральных процессоров в мульпроцессорных системах) распределяется между этими потоками.

Распределением времени центрального процессора занимается специальный модуль операционной системы - планировщик. Планировщик по очереди передает управление отдельным потокам, так что даже в однопроцессорной системе создается полная иллюзия параллельной работы запущенных потоков.

Следует особо отметить, что распределение времени выполняется для потоков, а не для процессов. Потоки, созданные разными процессами, конкурируют между собой за получение процессорного времени. Каждому потоку задается приоритет его выполнения, уровень которого определяет очередность выполнения того или иного потока.

2. Реализация многозадачности в Java

Для создания мультизадачных приложений Java необходимо воспользоваться классом java.lang.Thread. В этом классе определены все методы, необходимые для создания потоков, управления их состоянием и синхронизации.

Есть две возможности использования класса Thread.

Во-первых, можно создать собственный класс на базе класса Thread. При этом необходимо переопределить метод run(). Новая реализация этого метода будет работать в рамках отдельного потока.

Во-вторых, создаваемый класс, не являясь подклассом класса Thread, может реализовать интерфейс Runnable. При этом в рамка этого класса необходимо определить метод run(), который будет работать как отдельный поток.

Задания к лабораторной работе

Задание 1. Проверить и объяснить работу апплетов *MultiTask***,** *DoubleTask* и *Multi,*.

5. ПЕРЕЧЕНЬ ПРОГРАММНЫХ ПРОДУКТОВ, ИСПОЛЬЗУЕМЫХ В ПРЕПОДАВАНИИ ДИСЦИПЛИНЫ «История и методология информатики и вычислительной техники »

Лекции проводятся в стандартной аудитории, оснащенной в соответствии с требованиями преподавания теоретических дисциплин.

Для проведения лабораторных работ необходим компьютерный класс на 10 посадочных рабочих мест пользователей. В классе должны быть установлены Java.

6. ФОНД ТЕСТОВЫХ И КОНТРОЛЬНЫХ ЗАДАНИЙ ДЛЯ ОЦЕНКИ КАЧЕСТВА ЗНАНИЙ СТУДЕНТОВ

1. Что является главной целью науки:
a) получение знаний о реальности
b) развитие техники
c) совершенствование нравственности

2. Как называется метод получения эмпирического знания, при котором

главное - не вносить при исследовании какие-либо изменения в изучаемую реальность:
a) эксперимент
b) наблюдение
c) измерение

3. Как называется метод эмпирического познания, при котором изучаемое явление ставится в особые, специфические и варьируемые условия:
a) измерение
b) эксперимент
c) наблюдение

4. Язык науки является важнейшим средством научного познания. На каком языке, по утверждению Галилея, написана книга Природы:
a) математики
b) откровения
c) философии

5. Когда возникло естествознание?
a) в каменном веке, когда человек стал накапливать и передавать другим знания о мире;
b) примерно в V веке до н.э. в Древней Греции;
c) в период позднего средневековья XII-XIV вв.;
d) в XVI-XVII веках;
e) в конце XIX века.

6. Как называется тот структурный уровень науки, на котором знания являются результатом непосредственного контакта с "живой" реальностью в наблюдении или эксперименте:
a) эмпирический
b) теоретический
c) философский

7. Кто стал впервые широко применять мысленные эксперименты в ходе построения теории:
a) Ньютон
b) Галилей
c) Эйнштейн

8. Как называются научные теории, которые оперируют наиболее абстрактными идеальными объектами:
a) фундаментальные
b) теории конкретных явлений
c) общенаучные

9. Кроме эмпирического и теоретического в структуре научного знания можно выделить еще один уровень, содержащий общие представления о действительности и процессе познания. Какой это уровень:
a) философский
b) интерпретации
c) понимания

7. КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО – ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

№	Наименование дисциплин в соответствии с учебным планом	Обеспеченность преподавательским составом								
		Ф.И.О. должность по штатному расписанию	Какое образовательное учреждение профессионального образования окончил, специальность по диплому	Ученая степень и ученое звание	Стаж научно педагогической работы			Основное место работы, должность	Условия привлебчения к трудовой деятельности (штатный, совместитель, внутренний или внешний с указанием доли ставки), иное	Кол-во часов
					Всего	В т. ч. педагогический				
						Всего	В том числе по преподаваемой дисциплине			
	История и методология информатики и вычислительной техники	Самохвалова С.Г.	ДВГУ, математик	доцент, к.т.н.	24	20	1	АмГУ, каф. ИУС	Штатный	103