Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Горно-Алтайский государственный университет»

Н. Г. Кудрявцев, Д. В. Кудин, М. Ю. Беликова

ЭЛЕМЕНТАРНЫЕ ОСНОВЫ ИНФОРМАТИКИ

(Учебное пособие)

Горно-Алтайск
РИО Горно-Алтайского госуниверситета
2015

Печатается по решению редакционно-издательского совета
Горно-Алтайского государственного университета

УДК 004

ББК 32.973я73
К 88

Кудрявцев Н.Г., Кудин Д. В., Беликова М. Ю.
Элементарные основы информатики: учебное пособие / Н. Г. Кудрявцев, Д. В. Кудин, М. Ю. Беликова – Горно-Алтайск: РИО ГАГУ, 2015. – 100 с.

Рецензенты :

Власов В.Н., к. ф.-м. н., доцент кафедры общей информатики факультета информационных технологий Новосибирского государственного университета.

Кречетова С. Ю., к. ф.-м. н., доцент, заведующий кафедрой математики и информатики Горно-Алтайского государственного университета.

Пособие содержит краткий обзор основных разделов информатики и примеры решения задач по оценке количества информации, системам счисления и булевым функциям. Может использоваться при изучении дисциплин «Информатика», «Информатика и программирование» «Информационные технологии», «Информатика с основами биостатистики», «Информатика и современные информационные технологии», «Технология программирования и работа на ЭВМ», «Программирование».

© Кудрявцев Н.Г., Кудин Д. В., Беликова М. Ю., 2015

ОГЛАВЛЕНИЕ

1. Введение ... 5
2. Информация ... 5
2.1. Свойства информации ... 6
3. История возникновения науки информатики ... 7
4. Измерение информации ... 10
5. Кодирование визуальной информации ... 14
6. Представление звуковых данных ... 15
7. Представление текстовых данных ... 16
8. Модели окружающего мира. Моделирование ... 16
9. Понятие алгоритма ... 18
10. Множество. Операции над множествами ... 19
11. Графы ... 21
12. Конечные автоматы ... 23
13. Булевы функции ... 25
13.1. Способы задания булевых функций ... 25
13.2. Эквивалентные формулы ... 27
13.3. Специальные виды формул ... 28
14. История развития вычислительной техники ... 30
15. Элементная база компьютерной техники ... 48
16. Вычислительные устройства ... 50
16.1. Модель Фон-Неймана ... 51
16.2. Аппаратные средства компьютера ... 52
17. Программное обеспечение компьютера ... 54
17.1. Краткая история языков программирования ... 59
18. Системы управления базами данных ... 62
18.1. Принципы реляционной модели ... 63
18.2. Язык структурированных запросов SQL ... 65
19. Экспертные системы ... 65
19.1. Представление знаний ... 68
19.1.1. Семантические сети ... 69
19.1.2. Фреймы ... 69

19.1.3. Правила вывода 70
20. Открытые системы и сетевые технологии 70
20.1. Семиуровневая модель OSI/ISO 72
20.2. Передача данных по сети (функции пакетов) 77
20.3. Сетевое оборудование 77
20.4. Топологии локальных сетей 79
20.5. Система адресации в сети Internet 80
20.6. Доменная система имен 80
20.7. Порты и сокеты 81
21. Информационная безопасность 82
21.1. Правовые меры защиты информации 83
21.2. Организационные меры защиты информации 85
21.3. Технические средства защиты информации 87
22. Приложение 1. Задачи по измерению количества информации 91
23. Приложение 2. Набор карточек для самостоятельной работы 92
24. Приложение 3. Контрольные проверочные вопросы 95
25. Литература 100

1. Введение

Общество, в котором мы живем, все чаще называют информационным обществом и это – не пустые слова. Нас окружает огромное количество информации. За последние 35 лет выпущено столько же книг, что и за последние 500 лет. Сейчас выпускается около 350000 периодических изданий. Ежегодно публикуется около 10 млн. статей.

Сейчас невозможно усваивать большое количество информации. Сейчас важно научиться оперировать информацией, анализировать, сопоставлять, делать выводы, т.е. работать с информацией.

Уже в середине семидесятых годов больше половины работников в развитых Европейских странах и в США так или иначе были связаны с обработкой и анализом данных. В наши дни информационными потоками охвачены практически все промышленно развитые и развивающиеся страны мира. И это не только файловый и почтовый обмен через глобальную сеть Интернет. Это – финансовые операции, обеспечивающие международные платежи; это – потоки видео и звукового контента; это – массивы данных трансконтинентальных управляющих и измерительных систем; это –данные дизайнерских и конструкторских систем, это телефонные и видео переговоры, это запросы и результаты поисковых запросов в многочисленных базах данных и многое, многое другое.

Именно поэтому подготовка специалистов участвующих в процессах создания, передачи, хранения и поиска информации, специалистов, способных разрабатывать, эксплуатировать и внедрять информационные системы является актуальнейшей современной задачей для каждой развитой и развивающейся страны.

2. Информация

Информация – лат. *informatio* – «*Разъяснение*», «*Изложение*», наряду с энергией и материей – одно из понятий основ мироздания.

Информация в быту– сведения об окружающем мире и протекающих в нем процессах, воспринимаемых человеком или специальными устройствами.

Информация в технике – сообщения в форме знаков и сигналов.

Под информацией в теории информации понимаются только те сведения которые уменьшают или полностью снимают неопределенность (Шеннон: информация – мера снятой неопределенности). Интерпретация: мера – это минимальное количество бит, которыми можно закодировать каждое из сообщений

Рассмотрим еще несколько определений информации, позволяющих рассмотреть данное понятие с разных сторон.

Информация – обмен сигналами или сведениями, уменьшающими или снимающими неопределенность.

Информация – общенаучное понятие, включающее обмен сведениями между людьми, человеком и автоматом, автоматом и автоматом, обмен сигналами в животном и растительном мире, передачу признаков от клетке к клетке, от организма к организму.

Информация – сообщение, осведомление о положении дел, сведения о чем либо, передаваемые людьми; уменьшаемая, снимаемая неопределенность, в результате получения сообщения; отражение разнообразия в любых объектах и процессах живой и неживой природы.

Информация всегда фиксирует отличия и изменения. Поздно имеет смысл, так как существует рано. Создавая информацию, мы делаем цельное дискретным.

Информация – это особая совокупность сведений, первоначальным источником которой является опыт.

Информация – атрибут разумного существования; сведения, данные факты, полученные из опыта путем наблюдений, путем размышлений, зафиксированные в материальной форме для сообщения другими или самому себе.

Теория информации – формальные свойства информации (частотно–вероятностные, структурные аспекты), исследуемые математическими методами

Информацию связывают с разнообразием и степенью упорядоченности.

Информация – динамический объект, возникающий в момент взаимодействия объективных данных и субъективных методов.

При таком подходе информация содержит две компоненты: содержательную и материальную.

Информация должна быть:

– понятна (содержательная компонента – русский, английский язык, кодирование)
– представлена на материальном носителе (материальная компонента)

2.1. Свойства информации

– Объективность, полнота, достоверность, адекватность, доступность, актуальность;
– Объективность и субъективность – (фотографии – рисунки, исторические факты – юридические факты);

– Полнота – достаточность для принятия решения;
– Достоверность – отношение сигнал/шум, отношение полезного к бесполезному, если нужна большая достоверность, то нужно больше данных, либо более сложные методы для обработки данных;
– Адекватность – степень соответствия реальному состоянию дел;
– Доступность – мера возможности получить ту или иную информацию;
– Актуальность – степень соответствия текущему моменту.

Данные – зарегистрированные сигналы (при регистрации обычно происходит преобразованные сигналов из одного вида в другой).

Операции с данными:

– сбор данных – накопление информации;
– формализация – приведение к одинаковой форме (повышение уровня доступности);
– фильтрация – отсеивание лишних данных;
– сортировка – упорядочение по заданному признаку;
– архивация – организация хранения;
– защита – меры по предотвращению утраты, модификации, воспроизведения;
– транспортировка – прием – передача;
– преобразование – изменение типа носителя, представления и т.п.

3. История возникновения науки информатики

1834 г. Физик *Андре Мари Ампер* (1775 - 1836) предложил свою классификацию наук. Для науки об управлении обществом он предложил название *кибернетика* (греч. kubernētes — «рулевой», «кормчий», «искусство управления»).

Во второй половине сороковых годов XX века видный американский математик, известный своими работами по гармоническому анализу и исследованию случайных процессов, *Норберт Винер* (*Norbert Wiener*, 1894 — 1964) выдвинул новую концепцию управления как в технических, так и биологических и социальных системах. Работая над теорией сообщений, он отождествил понятия «коммуникация» и «управление». В своей книге «*Кибернетика*», вышедшей в 1948 г., он писал, что, для того чтобы установить связь с другим лицом, ему необходимо сообщить сигнал, а когда это лицо устанавливает связь с вами, то оно возвращает подобный сигнал, содержащий информацию, первоначально недоступную для вас. Если необходимо управлять действиями другого лица, то ему так же сообщается сигнал; и хотя этот сигнал дается в императивной форме, техника коммуникации не отличается от техники коммуникации при сообщении сигнала факта. Когда вы посылаете приказ машине, то возникающая в данном случае ситуация не отличается от ситуации,

когда вы отдаете приказ какому-либо лицу. В обоих случаях отдающий приказ посылает сигнал команды и получает сигнал подтверждения. В процессе изложения Н. Винер последовательно показывает, что теория управления в человеческом обществе, живой природе или механической технике является частью теории информации. Таким образом, было заявлено новое направление исследований – *кибернетика*. В интерпретации Н.Винера кибернетика рассматривалась как наука об управлении, получении, передаче и преобразовании информации в кибернетических системах. Важнейшая общая закономерность процессов управления, указанная Н. Винером, состоит в использовании обратной связи и передаче информации для реализации этой обратной связи.

Винер утверждал, что многие концептуальные схемы, определяющие поведение живых организмов при решении конкретных задач, практически идентичны схемам, характеризующим процессы управления в сложных технических системах. И более того, он был убежден, что социальные модели управления и модели управления в экономике могут быть проанализированы на основе тех же общих положений, которые разработаны в области управления системами, созданными людьми.

В своей программе развития кибернетики Винер поразительно точно спрогнозировал генеральный путь развития технологий. Он писал, что автоматизация производства потребует широкого использования компьютеров в системах управления и учета, а также значительного развития средств коммуникаций. И действительно, компьютер, рассчитывающий форму лопатки турбины и затем непосредственно управляющий изготовлением этой лопатки на станке, расположенном на другом берегу океана, стал реальностью уже в 60-е годы XX века. Что же касается телекоммуникаций, то они давно переросли рамки средств организации производства и стали важнейшим фактором социальной жизни.

Как же обстоит дело с кибернетикой сейчас? Прошли те времена, когда учебники и кафедры кибернетики были достаточно широко распространены. Теперь, в лучшем случае, можно встретить упоминание о кибернетике лишь в контексте со словом «экономическая». Уже в «предперестроечные» времена найти хотя бы одного «кибернетика» было невозможно! И если ученому, который занимался одной из проблем, поставленной Н. Винером в качестве предмета исследований кибернетики, задавался вопрос о специальности, то этот ученый, с большой долей вероятности, отвечал, что его специальность — прикладная математика, моделирование или программирование. Да и упомянутые учебники и кафедры в своем названии обычно содержали дополнительное уточнение: «техническая», «математическая» или «экономическая» кибернетика. Приходится признать, что и в нашей стране, и во всем мире «кибернетика вообще» как научная дисциплина так и не сложилась. Круг вопросов, отнесенных Н. Винером к кибернетике, оказался

разобранным множеством взаимосвязанных, но все-таки особых научных дисциплин: теорией оптимального управления, теорией распознавания образов, теорией передачи информации, исследованием операций, искусственным интеллектом, системным программированием.

Отметим ниже три важных момента, связанные с теорией информации, на которые обращал внимание в своих работах Н.Винер:

- связь - это процессы восприятия, хранения и передачи информации;
- управление - это процесс воздействия управляющей системы на объект управления, обеспечивающей его работу по достижению заданной цели;
- управление осуществляется с помощью сигналов, которые содержат информацию об управлении.

В 1956 г. *Росс Эшби* (1903-1972) систематизировал идеи Винера. В его работах также отмечалось, что Кибернетика изучает способность машин и животных организовывать, воспринимать, определенную информацию, сохранять ее в памяти, передавать ее по каналам связи и перерабатывать в управляющие сигналы. К задачам кибернетики было отнесено - выделение, изучение, обобщение свойств процессов управления и систем управления.

Кибернетику делили на несколько ветвей:

- *теоретическая кибернетика* – математическая логика, теория алгоритмов, теория информации, теория кодирования, теория автоматов, теория формальных языков и грамматик, нечеткие логики, математические модели, компьютерные модели;
- *техническая кибернетика* – исследование всевозможных технических управляющих систем, теория автоматического управления
- *биологическая кибернетика* – медицина, биология, физиология, математические модели, математическая статистическая обработка.

По мере развития кибернетики и расширении спектра задач, которые решались кибернетическими методами, все чаще стали использоваться компьютеры. В основном они были нужны для обработки математических моделей. Соответственно стали развиваться языки программирования. Кибернетика не давала ответы на вопросы, связанные с развитием вычислительной техники. В 40-е 50 - е годы XX-века стало формироваться направление кибернетики, которое в англоязычных странах назвали *«Computer Science»* Во Франции назвали *Informatique*, объединив слова *Information* и *Automatique*. Постепенно к Информатике перешли все вопросы, связанные с созданием, хранением, передачей и преобразованием информации. У кибернетики остались только проблемы управления. На западе этот круг обозначается как «*Computer Science*», а у нас все чаще и чаще используется слово «*информатика»*.

Еще один вариант определения *информатики* – это наука о том как создается, хранится и перерабатывается информация, где под информацией понимаются данные, знания, факты.

Сейчас к информатике относят следующие направления научных исследований: программные системы; высокопроизводительные системы; сетевые технологии; дискретную математику; вычислительную математику (решение задач с помощью компьютеров); системный анализ (процессы манипулирования данными); автоматизацию процессов передачи и обработки данных; инженерные аспекты конструирования устройств и программного обеспечения.

Информатику иногда делят на экономическую, правовую, медицинскую, в зависимости от сферы применения ЭВМ, баз данных и применения прикладных программ.

К разделам информатики обычно относят: алгоритмы и структуры данных, архитектуру и организацию компьютеров, языки и программы, операционные системы, технологию программирования, взаимодействие человека и компьютера и т.п.

Теперь перейдем к разделу, описывающему некоторые общепринятые подходы к измерению количества информации, содержащейся в определенном сообщении.

4. Измерение информации

Ученые пришли к соглашению называть, минимальное количество информации, которое может быть передано или получено *1 битом (bit - binary digit).* Также договорились считать, что вы получаете 1 бит информации, если сообщение, адресованное вам, содержит сведения о том какое из двух равновероятных событий имело место быть.

Теперь поговорим о том каким образом измерять информацию, содержащуюся в произвольном сообщении. Сначала введем понятие вероятности наступления какого-то события. Вероятность – отношение количества наблюдений, при котором нужное нам событие наступило к общему количеству наблюдений.

В 1928 году *Хартли* предложил свою формулу для определения количества информации, если есть N равновероятных событий, то информация о каком то событии вычисляется по формуле $I = log_2 N$ Другими словами, если вероятность события $p = \frac{1}{N}$, то информация о наступлении этого события будет равна: $I = log_2 \frac{1}{p}$ Можно сказать, что, чем менее вероятное событие произошло, тем больше информации содержится в сообщении о том, что оно случилось.

Пример 4.1: У вас есть колода карт из 36 карт, то вероятность вытащить даму пик 1/36, тогда, исходя из формулы Хартли, делаем вывод о

том, что сообщение о вытащенной даме пик содержит: $I = log_2 36 = \frac{lg36}{lg2} = \frac{1.5563}{0.301} = 5.17$ *бит информации. В то же самое время сообщение о том, что была вытащена карта червовой масти содержит всего* $I = log_2 4 = 2$ *бита информации.*

Другими словами, можно сказать, что, если существует *N* равновероятных исходов какого-то эксперимента, то количество информации в сообщении о результатах данного эксперимента равно количеству вопросов (ответ на которые да/нет), необходимых, для того чтобы идентифицировать (угадать) искомый исход. (Каждый вопрос сокращает выбор в 2 раза)

Для *N* вариантов количество бит $k = log_2 N$ *(формула Хартли)*

Для неравнозначных вариантов $k = - log_2 p$, где – p вероятность искомого события.

Клод Элвуд Шеннон (1916-2001) в математической теории передачи информации ввел понятие энтропии, которую можно вычислить по формуле

$$H = \sum_{i=1}^{N} -p(x_i) log_2 p(x_i)$$

Эта формула позволяет определить минимальное количество бит, которое необходимо для того, чтобы закодировать любое из *N* неравновероятных сообщений, если использовать кодовые слова различной длины, обратно пропорционально вероятности встречаемости кодируемого символа (кодирование по *Хаффману*). Также энтропию называют оптимальной или средней длиной кодового слова.

Пример 4.2: За два года ученик получил всего 100 оценок: 60 пятерок, 25 четверок, 10 троек, 5 двоек. Необходимо определить оптимальную длину кодового слова при оптимальном кодировании (по Хаффману). Сразу отметим, что при равновероятных исходах получения оценок сообщение о получении любой оценке будет содержать 2 бита информации, а для кодирования сообщения о 100 полученных оценках необходимо будет использовать 200 битную кодовую последовательность.

Теперь оценим длину оптимального кодового слова по формуле Шеннона.

$$H = \sum_{i=1}^{N} -p(x_i) log_2 p(x_i) = \frac{60}{100} log_2 \frac{100}{60} + \frac{25}{100} log_2 \frac{100}{25} + \frac{10}{100} log_2 \frac{100}{10} + \frac{5}{100} log_2 \frac{100}{5}$$

$$H = 0.6(log_2 5 - log_2 3) + 0.25 * 2 + 0.1 log_2 10 + 0.05 log_2 20$$

$$H = 0.442 + 0.5 + 0.332 + 0.216 = 1.49$$

Оптимальное кодовое слово будет содержать 1.49 бит информации. Это значит, что для передачи сообщения о 100 полученных оценках при оптимальном кодировании будет использована кодовая последовательность размером не менее 149 бит.

Азбука Морзе - наиболее часто встречающиеся буквы кодируются наиболее короткими последовательностями - один из примеров оптимального кодирования.

Средняя длина кодовых слов: $E(L)= p_1L_1 + p_2L_2+ ...+ p_nL_n$, где L_i длина i-ого кодового слова

Первая теорема Шеннона для кодирования канала без шума: Средняя длина кодового слова E(L) не меньше энтропии.

E(L) - H(L) = r(L) - избыточность кодирования.

В процессе преобразования, хранения и передачи информации часто используются различные способы кодирования или представления информации.

Процессор понимает команды в двоичном виде. При написании управляющих программ эти же команды могут иметь как десятичное, восьмеричное, так и шестнадцатеричное представление.

Любое из перечисленных выше представлений чисел относится к позиционным системам счисления. Основание позиционной системы счисления определяет возможное количество цифр в любом из разрядов. Каждый разряд характеризуется своим весом – степенью основания системы счисления. Чем больше основание системы счисления, тем меньшим количеством разрядов может быть представлено число. В вычислительной технике используются двоичная, восьмеричная и шестнадцатеричная системы счисления. Преобразование из десятичной в любую из этих систем можно осуществить путем последовательного деления исходного числа на основание системы и выписывания остатков от деления на каждом промежуточном шаге.

Пример 4.3: Необходимо преобразовать десятичное число 28 к двоичному виду.

На первом шаге находим максимальную степень двойки меньшую 28 – это 2^4, далее отнимаем от искомого числа найденную нами степень двойки 28-16=12. Теперь для числа 12 проделываем то же самое. Максимальная степень двойки меньшая двенадцати – это 2^3, разность: 12-8=4. Четыре, в свою очередь, это 2^2. Итак, имеем, что $28=2^4+2^3+2^2$. То же само равенство можно расписать через сумму всех степеней двойки (начиная со старшей), умноженных на 0 или 1. Значение 0 или 1 коэффициента при каждой степени двойки будет зависеть от того входит данная степень в разложение искомого числа или нет. Получаем: $28= 1*2^4+1*2^3+1*2^2+0*2^1+0*2^0$. Теперь, если мы возьмет коэффициенты при степенях двойки в том же порядке, как они у нас и записаны, то получим двоичное представление десятичного числа: $28_{10}=11100_2$. Значение нижнего индекса в последней записи указывает на систему счисления, в которой записано число.

К десятичному виду любое число (число, записанное в позиционной системе счисления с любым основанием) можно привести путем умножения каждого разряда представляемого числа на соответствующую его позиции степень основания системы счисления с последующим сложением получившихся произведений.

$$N = \sum_{i=0}^{k} a_i * b^i$$

В данной формуле a_i – значение i-ого разряда в преобразуемом числе (нумерация ведется справа налево).

Пример 4.4: $110100_2 = 1*2^5+1*2^4+0*2^3+1*2^2+0*2^1+0*2^0=32+16+4=52_{10}$

Пример 4.5: $324_8=3*8^2+2*8^1+4*8^0=3*64+2*8+4=192+16+4=212_{10}$

Преобразования между двоичной и восьмеричной системами счисления производят триадами двоичных цифр. Выбирают эквивалентную трем двоичным цифрам восьмеричную цифру, и производят замену в соответствующей позиции преобразуемого числа.

Пример 4.6: $010\ 111\ 011_2=273_8$

Преобразования между двоичной и шестнадцатеричной системами производят квадрами (четверками) двоичных цифр.

Пример 4.7: $1011\ 1011_2=BB_{16}$

Выполнение арифметических операций имеет смысл рассматривать только на примере двоичной системы счисления, так как именно такие операции реализуются аппаратно (в компьютере).

Сложение производится так же, как и в десятичной системе счисления с переносом в старший разряд.

Пример 4.8: 10011 *Проверка: 19+14=33*

$$\begin{array}{r} 10011 \\ \underline{1110} \\ 100001 \end{array}$$

Умножение на 2 реализуется сдвигом влево (дописыванием нуля справа).

Пример 4.9: 11 -> 110 *Проверка: 3*2=6*

Деление на 2 реализуется сдвигом вправо (исключением одного разряда справа).

Пример 4.10: 1000 -> 100 *Проверка: 8/2=4*

Вычитание можно выполнить путем преобразования в дополнительный код вычитаемого, сложение уменьшаемого и вычитаемого и отбрасывания старшего разряда.

Пример 4.11: Найти разность двух двоичных чисел 100110_2 - 10110_2.

Для нахождения разности, вычитаемое представляют в дополнительном коде (инверсией всех разрядов и прибавлением единицы к младшему разряду) 010110-> 101001+1 -> 101010,

затем складывают уменьшаемое и дополнительный код вычитаемого 100110 и отбрасывают старший разряд.

100110
101010
1010000
Проверка 38-22=16

5. Кодирование визуальной информации

В 1756 году *Михаил Васильевич Ломоносов (1711-1765)* высказал мысль о том, что для воспроизведения любого цвета достаточно смешения трех основных цветов.

Растровые изображения. Картинка по вертикали и горизонтали разбивается на маленькие прямоугольники (растр). Элементы растра - *пикселы (picture's elevtynt*). Цвет каждого пикселя определяется тройкой значений интенсивности (RGB - Red, Green, Blue) (Красный, Зеленый, Синий) в аддитивной модели. На *Рис.5.1* показаны сочетания цветов.

Сочетания цветов:

Magenta (Пурпурный, Сиреневый) = Красный + Синий;
Yellow (Желтый) = Красный +Зеленый;
Cyan (Циановый, Голубой) = Синий +Зеленый
Белый = Красный + Синий + Зеленый

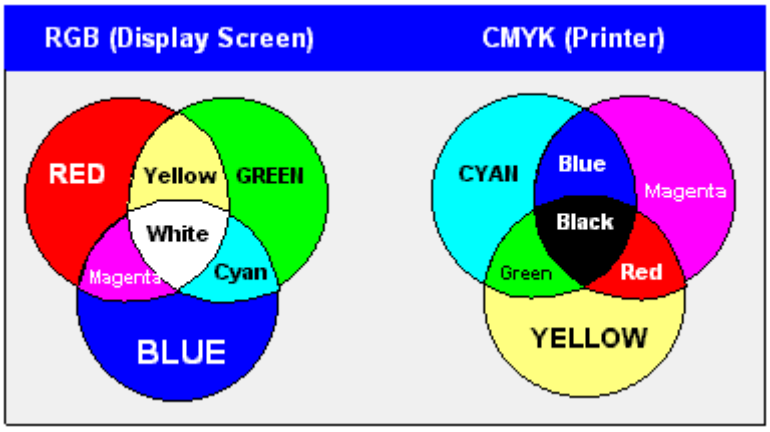


Рис. 5.1

При печати свет не излучается, а отражается от бумаги. Из-за оптических свойств отраженного светового сигнала в печатной промышленности принята модель CMYK, где базовыми уже являются не красный, зеленый и синий цвета, а голубой, сиреневый, и желтый. Хотя черный цвет и можно получить путем

смешивания голубого, сиреневого и желтого цветов, он вводится в палитру отдельным цветом и обозначается отдельной буквой К, так как при печати используется в больших количествах и для его получения путем смешивания уйдет очень много краски других цветов.

Также наряду с аддитивными схемами RGB и CMYK существует система HSV

Hue - цветовой тон
Saturation - насыщенность
Value - яркость, уровень излучения

Чем больше бит отводится на каждый цвет, тем большую гамму цветов можно хранить. Стандартно для каждого цвета отводится по одному байту (всего 24 бита). Таким образом можно определять 256 уровней яркости на каждый цвет. Итого 16.7 млн. различных оттенков.

Векторные изображения. Векторные изображения в отличии от растровых хранят информацию не об отдельных пикселях, а об элементарных геометрических фигурах (примитивах: точках, прямых, треугольниках, прямоугольниках, окружностях, овалах и т.п). В конечном итоге картинка строится из большого количества геометрических примитивов. Очевидно, что адекватную картину природы таким образом построить вряд ли получится, а вот какой-то чертеж детали, проект дома, карту местности получить относительно просто. Основное преимущество векторных изображений перед растровыми — их масштабируемость. При увеличении масштаба у простой растровой фотографии в конечном итоге появляются размытости и качество резко снижается. Если же надо получить увеличенное изображение или какой-то необычный ракурс части чертежа детали, то это можно сделать простым пересчетом пропорций, практически без потери качества.

6. Представление звуковых данных

Звук – это колебания – изменения амплитуды воздуха (звуковые волны).

Звуковые волны, равно как и все остальные, характеризуются частотой повторения (Гц -количество колебаний в секунду), амплитудой и фазой.

Процесс представления звука в компьютере состоит из нескольких шагов. Сначала выполняется преобразование звуковых волн в электрический ток (при помощи разного рода микрофонов). Далее происходит преобразование непрерывного электрического сигнала в двоичный – дискретизация (аналого-цифровое преобразование). Затем осуществляется преобразование сигнала с последующим сохранением (усиление, фильтрация). Воспроизведение путем цифро-аналогового преобразования является последней частью в череде преобразований. Нас будет интересовать процесс дискретизации, качество которого можно оценить двумя параметрами: частотой дискретизации *bitrate*– количеством измерений в секунду и разрядностью преобразователя

(разрешение) в битах. От этих же показателей и зависит объем записанного в памяти компьютера звукового файла.

Пример 6.1: какой объем в килобайтах будет занимать звуковой файл, содержащий минуту звуковой записи, выполненной с частотой дискретизации 44032 Гц при помощи 16 разрядного АЦП.

*Решение: 1 секунда такой записи занимает 44032 * 16=704512 бит или 88064 байта, соответственно для минуты такой записи потребуется 88064 * 60 = 5283840 байт. Это 5160 килобайт или 5,0390625 мегабайт*

7. Представление текстовых данных

Простейший вариант представления текстовых данных осуществляется путем сопоставления каждой букве, каждому символу однобайтового числа (кода). Это так называемая *ASCII* кодировка. *ASCII - American Standart Code for Information Interchange*

Пример 7.1: Книга содержит 150 страниц, каждая страница - 40 строк, в каждой строке 60 символов. Каков объем файла, требуемого для хранения данной книги, если каждый символ кодируется в ASCII кодировки.

*Решение: 1 символ - 1 байт. 1 страница соответствует объему 40 * 60 = 2400 байт. Вся книга, содержащая 150 страниц занимает объем 2400 * 150 = 360000 байт*

360 000 /1024 = 351,5625 Кбайт

351,5624 /1024 = 0,34332275 Мбайт

Пример 7.2: Объем Дюма "Три мушкетера" 590 страниц, на каждой странице 48 строк, в каждой строке 53 символа. Вычислить объем файла, который требуется для хранения данной книги.

*Решение: 53 * 48 * 590 = 1500 960 символов*

1500960 / 1024 = 1465,78125 Кбайт

1465,78125 / 1024 = 1,431427 Мбайт

8. Модели окружающего мира. Моделирование

Решая задачи управления, в реальной жизни почти всегда приходится сталкиваться со сложными объектами, в описании которых часто используют понятие системы.

Термин «*Система*» используют в тех случаях, когда хотят охарактеризовать исследуемый или проектируемый объект как нечто целое (единое) сложное, о котором невозможно сразу дать представление. Например, «система управления станком», «система кровообращения» и т.д. В математике термин система используется для отображения совокупности математических выражений или правил. Понятие системы подчеркивает упорядоченность, целостность, наличие определенных закономерностей.

«*Система*» – это совокупность элементов, находящихся в отношениях и связях друг с другом, которые образуют определенное целостное единство. Целостность при этом рассматривается как принципиальная несводимость свойств системы к сумме свойств составляющих ее элементов, и не выводимость из свойств последних целого; зависимость каждого элемента, свойства и отношения системы от его места, функции и т.д. внутри целого.

Что же такое моделирование систем или просто моделирование? При большом количестве определений понятие «модель» сводится к следующему: модель – это специально подобранный объект (причем объект, как реальной, так и виртуальной, абстрактной природы), который имеет с моделируемым объектом некоторые общие свойства, интересующие исследователя.

Одно из определений, достаточно точно отражающее суть проблемы, дал С. Лем: «Моделирование – это подражание природе, учитывающее немногие ее свойства». «Моделируя, следует упрощать».

Соотношение реального объекта и модели можно представить следующим образом (Рис.8.1):

Рис.8.1 Соотношение реального объекта и модели

В некоторых случаях предлагают делить модели на натуральные и знаковые. Натуральная модель – это реальный (физический, биологический, химический и др.) объект, характеристики которого изменяются по тем же законам, что и показатели исследуемого объекта. Знаковая модель состоит из графических объектов (схемы, графики, символы, формулы и т.д.), связываемых определенными правилами и преобразованиями».

Достаточно большая часть знаковых моделей являются математическими моделями. *Математическая модель* – приближенное описание какого-либо класса явлений внешнего мира, выраженное с помощью математической символики.

Процесс математического моделирования, т.е. изучения явления с помощью математической модели, подразделяют на 4 этапа.

Первый этап построения модели заключается в формулировании законов, связывающих основные объекты модели и завершается записью в математических терминах сформулированных качественных представлений о связях между объектами модели. Разрабатывается или выбирается знаковая

система - язык моделирования. В качестве языка моделирования может использоваться естественный язык, средства теоретико-множественных, логических, лингвистических и др. методов дискретной математики. По мере развития процесса постепенной формализации язык моделирования может изменяться.

Второй этап - исследование математических задач, к которым приводят математические модели. Основным вопросом здесь является решение прямой задачи, т.е. получение в результате анализа модели выходных данных (теоретических следствий) для дальнейшего их сопоставления с результатами наблюдений изучаемых явлений. Выбирается подход к моделированию систем, и вводятся правила преобразования, применяемые при формировании и анализе модели правила структуризации или декомпозиции (подход «сверху»); правила композиции, поиска мер близости на пространстве состояний элементов (подход «снизу»).

Третий этап – выяснение того, удовлетворяет ли принятая (гипотетическая) модель критерию практики, т.е. выяснение вопроса о том, согласуются ли результаты наблюдений с теоретическими следствиями в пределах точности наблюдений.

Четвертый этап – последующий анализ модели в связи с накоплением данных об изучаемых явлениях и модернизация модели. Если наступает момент, когда выводы, получаемые на основании принятой математической модели, не соответствуют нашим знаниям о явлении, то возникает необходимость построения новой, более совершенной математической модели.

Используя различные математические модели, пытаются исследовать различные явления окружающего мира. В большинстве случаев эти явления исследуются не для удовлетворения праздного любопытства, а для нахождения закономерностей, которым эти явления соответствуют и подчиняются. Когда же такие закономерности найдены, приступают к решению практических задач. Пути достижения решения той или иной задачи обычно описывают в виде определенных последовательностей действий или алгоритмов.

9. Понятие алгоритма

Алгори́тм, от имени учёного *аль-Хорезми* – точный набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время. В старой трактовке вместо слова «порядок» использовалось слово «*последовательность*», но по мере развития параллельности в работе компьютеров слово «*последовательность*» стали заменять более общим словом «*порядок*».

Ранее часто писали *«алгорифм»*, сейчас такое написание используется редко.

Различные определения алгоритма в явной или неявной форме содержат следующий ряд общих требований:

- *Дискретность* – алгоритм должен представлять процесс решения задачи как последовательное выполнение некоторых простых шагов. При этом для выполнения каждого шага алгоритма требуется конечный отрезок времени, то есть преобразование исходных данных в результат осуществляется во времени дискретно.
- *Детерминированность* (определённость). В каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Таким образом, алгоритм выдаёт один и тот же результат (ответ) для одних и тех же исходных данных.
- *Понятность* – алгоритм должен включать только те команды, которые входят в систему команд исполнителя.
- *Завершаемость* (конечность) – при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов.
- *Массовость* (универсальность). Алгоритм должен быть применим к разным наборам исходных данных.
- *Результативность* – завершение алгоритма определёнными результатами.

Алгоритмы обычно принято изображать при помощи блок-схем алгоритмов *(БСА)* – набора связанных стрелками геометрических фигур, обозначающих определенные действия в программе (параллелограмм – загрузку/выгрузку данных, ромб-проверку условия, прямоугольник– выполнение одного или нескольких операторов), либо посредством псевдокода (подобие программы, где в каждой строке словами описываются определенные действия)

До 40-50-х годов прошлого века реализация как вычислительных алгоритмов, так и алгоритмов управления осуществлялась при помощи механических или электромеханических устройств. При современном уровне развития цифровой техники подавляющее большинство (практически все) алгоритмы реализуются при помощи цифровых устройств. Для того чтобы понять основные принципы работы современных цифровых устройств, которые лежат в основе современной вычислительной техники необходимо рассмотреть несколько базовых моделей из области дискретной математики.

10. Множество. Операции над множествами

Понятие *множества*, как и любое другое исходное понятие математической теории, не определяется, поскольку всякое определение ссылается на уже используемые понятия, логически предшествующие определяемому. Множеству трудно дать какое либо определение, которое не

сводилось бы к замене слова «*множество*» его синонимами: набор, совокупность, собрание, элементов.

В качестве рабочего определения понятия множество можно предложить следующее: *Множество* – это некоторая неупорядоченная совокупность неповторяющихся элементов, объединенных по определенному признаку или удовлетворяющая определенному свойству (набору свойств).

Множества мы будем обозначать прописными буквами $A, B...$ а их элементы - малыми $a, b, c...$. Утверждение «элемент a принадлежит множеству A» символически записывается так: $a \in A$. Запись $a \notin A$ означает, что a не принадлежит A. Если все элементы, из которых состоит A входят и в B, причем случай $A = B$ исключается, т.е., имеет место строгое вхождение, то мы называем A подмножеством множества B и пишем $A \subset B$. Например, целые числа образуют подмножество в множестве действительных чисел, или множество манипуляторов является подмножеством множества всех технических систем, развернутых в мастерской. Если в мастерской имеются только манипуляторы, то можно сказать, что имеет место нестрогое вхождение, и в этом случае пишут $A \subseteq B$.

Иногда мы не знаем заранее, содержит ли некоторое множество, например множество шагающих тележек, находящихся в данный момент в лаборатории, хотя бы один элемент. Поэтому целесообразно ввести понятие пустого множества, т.е. множества не содержащего ни одного элемента. Мы будем обозначать его символом $\varnothing$. Любое множество содержит $\varnothing$ в качестве своего элемента. Подмножества некоторого множества отличные от него самого и от $\varnothing$ называются собственными.

Рассмотрим пример *10.1*

$A = \{a_1, a_2, a_3, a_4, a_5, a_6, \{a_7, a_8\}\}$

$B = \{a_1, a_2, a_6\}$

$C = \{a_7, a_8\}$

Для множеств A, B и C можно записать следующие соотношения.

$B \subset A$

$a_2 \in A, a_7 \notin A$

$C \in A$

Мощностью множества A будем называть количество элементов этого множества, и обозначать, как |A|.

Пусть A и B - некоторые множества; их суммой или объединением $C = A \cup B$ называется множество, состоящее из элементов, принадлежащих хотя бы одному множеству. Аналогично определяются объединение произвольной, в том числе и бесконечной совокупности множеств.

Назовем пересечением $C = A \cap B$ множеств A и B множество, состоящее из всех элементов, как A, так и B.

Разностью $C = A \setminus B$ двух множеств A и B определим совокупность тех элементов из A, которые не содержатся в B, при этом не предполагается, что $B \subset A$

Симметрическая разность определяется как сумма разностей $A \setminus B$ и $B \setminus A$. Таким образом, по определению: $A \therefore B = (A \setminus B) \cup (B \setminus A)$

11. Графы

Графы в математике используются для исследования отношений между объектами, для наглядного представления этих отношений и взаимозависимостей. Также при помощи графов исследуют транспортные задачи, строят сетевые графики, обозначают переходы между состояниями конечных автоматов и т.п. Исследуемые объекты разной природы обозначают кружочками или точками и называют вершинами графа, отношения между исследуемыми объектами обозначают линиями или стрелочками и называют ребрами или дугами.

Формально, для того, чтобы задать граф, необходимо задать множество вершин $V=\{v_1,v_2,...,v_n\}$ и множество отношений (множество пар) на этом множестве вершин: $E=\{(v_1v_2), (v_2v_1), (v_1v_4),...,(v_kv_m)\}$. Каждую пару из множества отношений называют дугой.

В зависимости от того, как задано множество отношений различают три типа графов.

Если на множестве отношений для каждой пары (v_iv_j) существует противоположная ей пара (v_jv_i), то в графическом представлении такого графа между вершинами (v_iv_j) проводят простые линии, обозначающие симметричность отношений, а такой граф называют неориентированным.

Если на множестве отношений существуют пары (v_iv_j), для которых противоположная ей пара (v_jv_i) не задана, то в графическом представлении такого графа между вершинами (v_iv_j) проводят стрелки, обозначающие асимметричность отношений, а такой граф называют смешанным.

Если на множестве отношений не существует ни одной пары (v_iv_j), для которой была бы задана противоположная ей пара (v_jv_i), то в графическом представлении такого графа между вершинами (v_iv_j) проводят стрелки, обозначающие асимметричность отношений, а такой граф называют ориентированным.

Пример 11.1: Зададим граф $G = \{V, E\}$

$V = \{v_1, v_2, v_3, v_4, v_5\}$; $E = \{(v_1v_2), (v_1v_3), (v_3v_3), (v_2v_5), (v_2v_4), (v_3v_4), (v_4v_5)\}$ *и изобразим его на Рис.11.1*

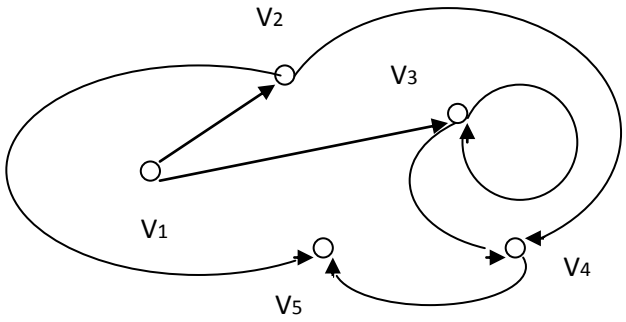


Рис.11.1

Существует несколько способов задания графов:

Теоретико-множественный – путем задания множества вершин и множества отношений на множестве вершин (как это было показано выше).

Графический способ - путем рисования множества вершин и множества соединяющих ребер или дуг.

Путем задания матрицы смежности или матрицы инциденций

Матрицей смежности называют матрицу размерности *n x n*, где *n* – число вершин графа. Элемент матрицы a_{ij} равен количеству дуг, идущих из вершины v_i в вершину v_j. Если элемент a_{ij} не равен нулю, то вершины *i и j* являются смежными. Если матрица симметрична относительно главной диагонали, то граф, который описывает данная матрица - неориентирован.

Рассмотрим матрицу смежности (Рис.11.2) графа *G={V,E}* из предыдущего примера

	v_1	v_2	v_3	v_4	v_5
v_1		1	1		
v_2				1	1
v_3			1	1	
v_4					
v_5				1	

Рис.11.2

Матрицей инциденций назовем матрицу размерности *n x m*. Строки этой матрицы соответствуют вершинам графа, а столбцы – его дугам. Элемент матрицы $a_{ij} = 1$, если ребро e_j выходит из вершины v_i, т.е. вершина является началом дуги; $a_{ij} = -1$, если ребро e_j входит в вершину v_i , т.е. вершина является коном дуги. Если граф с петлями, то матрицу инциденций расщепляют на две – матрицу начал дуг, и матрицу концов дуг. Рассмотрим матрицу инциденций (Рис.11.3) графа из предыдущего примера.

$A'(G) =$

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
v_1	1	1					
v_2			1	1			
v_3					1		1
v_4							
v_5						1	

$A''(G) =$

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
v_1							
v_2	-1						
v_3		-1			-1		
v_4				-1		-1	-1
v_5			-1				

Рис.11.3

12. Конечные автоматы

Автоматное представление алгоритмов и протоколов может существенно облегчить жизнь инженеров и программистов. Для того, чтобы разобраться в сути этого представления несколько слов скажем об автоматах и способах их описания.

Итак, *конечный автомат* – это пятерка, состоящая из трех множеств – множества входных воздействий, множества состояний, множества выходных реакций и двух функций переходов: между состояниями (в зависимости от текущих входных воздействий и предыдущего состояния) и между реакциями (также в зависимости от текущих входных воздействий и предыдущего состояния). Данное определение конечного автомата в основном используют при решении задач разработки и моделирования цифровых устройств. Определенный выше автомат принято называть автоматом *Мили.* Также при решении многих задач используются автоматы, у которых выходные реакции полностью определяются текущим состоянием, например триггеры. Такие автоматы называют автоматами *Мура*.

Выше мы уже говорили о том, что появление той или иной математической модели определяется необходимостью ее использования для решения определенного класса задач. Еще одно определение конечного автомата обусловлено применением данного класса моделей в синтаксических и лексических анализаторах а также при тестировании программного обеспечения.

В этом случае в определение конечного автомата включается понятие начального состояния q_0 и множества финальных (заключительных) состояний,

а множество реакций и функция определяющая эти реакции из определения автомата исключаются.

Определенный таким образом конечный автомат начинает работу в состоянии q_0, считывая по одному символу входной цепочки. Считанный символ переводит автомат в новое состояние в соответствии с функцией переходов. Читая на каждом такте своей работы входную цепочку *x*, автомат, после того как он прочитает последнюю букву цепочки, оказывается в каком-то состоянии *q*. Если это состояние является заключительным, то говорят, что автомат допустил цепочку *x*.

Классическое понятие конечного автомата предполагает однозначное задание как функций переходов между состояниями автомата, так и функций, определяющих его реакции. В этом случае принято говорить о детерминированных конечных автоматах (ДКА). Однако, для упрощения моделирования определенного класса задач используют понятие недетерминированного конечного автомата (НКА), у которого одно из основных отличий от ДКА заключается в том что функция переходов определяет в качестве следующего не одно какое-то состояние, а некоторое множество состояний. Таким образом, НКА может находится одновременно в нескольких состояниях. При этом доказывается, что каждому НКА можно сопоставить ДКА (имеющий, как правило, более сложную структуру).

Если переходы между состояниями конечного автомата осуществляются не однозначно, а с некоторой вероятностью, то говорят о вероятностных конечных автоматах.

Конечные автоматы можно задавать и описывать различными способами. Выделяют два основных способа: табличный (Таблица 12.1 – таблица переходов: строки – множество воздействий, столбцы – множество состояний, в клетках новые состояния и реакции) и графовый (Рис.12.1 – вершины графа – состояния, дуги – переходы между состояниями). Над дугами обычно подписывают условия перехода (входные воздействия и реакции, которые осуществляются при переходе).

Таблица 12.1

	Z_1	Z_2	Z_3
X_1	Z_2/Y_2	Z_3/Y_1	Z_3/Y_0
X_2	Z_3/Y_2	Z_2/Y_0	Z_3/Y_0
X_3	Z_1/Y_0	Z_2/Y_0	Z_2/Y_1

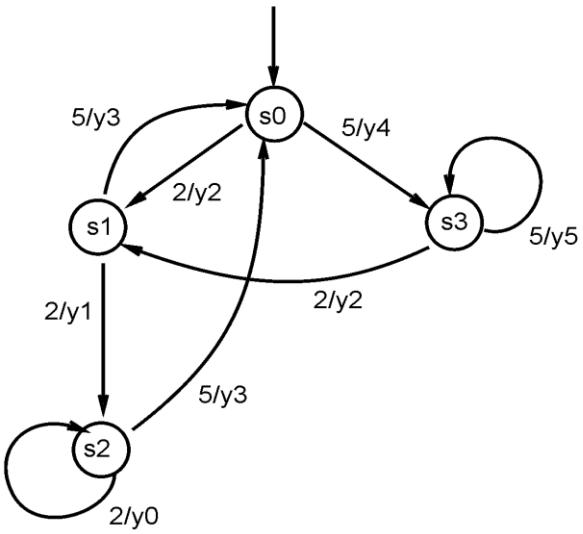


Рис. 12.1

13. Булевы функции

Функции вида *f:* E_2^n *->*E_2, где E_2*={0,1}* называются функциями алгебры логики или булевыми функциями. Другими словами, это функциональное соответствие между множеством кортежей (наборов или векторов) размерности *n* из *0,1* и множеством *{0,1}*. Всего существует 2^{2^n} булевых функций от *n* переменных.

13.1. Способы задания булевых функций

Булеву функцию можно задать таблицей истинности. Таблица истинности булевой функции от *n* переменных – таблица, состоящая из 2^n строк и *n+1* столбцов. Каждая строка представляет собой двоичный кортеж (вектор) размерности *n* и сопоставленное ему значение функции *(0 или 1)*. Все кортежи расположены в лексикографическом порядке.

Пример 13.1: Приведем пример таблицы истинности функции от трех переменных

x_3	x_2	x_1	y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Для упрощения и экономии места таблицу истинности можно заменить двоичным вектором размерности 2^n. Приведенную выше функцию можно записать в следующем виде *(11010010)*

Также булеву функцию можно задавать вектором единиц: $f(X)|_1 = \vee(0,1,3,6)$. Числа в скобках указывают на порядковые номера строк в таблице истинности, которые имеют единичные значения.

Элементарными булевыми функциями называются все функции одной переменной и некоторые функции двух переменных. Каждая элементарная функция имеет свое название и обозначение. Запишем в комбинированную таблицу истинности (Таблица 13.1) все функции двух переменных

Таблица 13.1

x_2	x_1	φ_0	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	φ_7	φ_8	φ_9	φ_{10}	φ_{11}	φ_{12}	φ_{13}	φ_{14}	φ_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Среди элементарных булевых функций можно выделить тривиальные функции (тождественно ложные или тождественно истинные). Если на всех наборах функция принимает значение 0 или 1, то такие функции называют константами 0 или константами 1 или тождественно ложными (тождественно истинными). Это функции φ_0 и φ_{15}

Еще в один класс элементарных булевых функций выделим все функции одной переменной x и $\bar{x}$. Это функции φ_5 и φ_{10}, φ_3 и φ_{12}

Следующий класс объединит некоторые функции двух переменных.

Логическое *И (КОНЪЮНКЦИЯ).* Функция, которая принимает единичные значения только на тех наборах, на которых и переменная x_1, и переменная x_2 равны единице. Функция коммутативна. Для обозначения функции используют следующие функциональные символы:

$$\varphi_1 = x_1 \,\&\, x_2 = x_1 \wedge x_2 = x_1 \cdot x_2 = x_1 x_2$$

Логическое *ИЛИ (ДИЗЪЮНКЦИЯ*). Функция, которая принимает единичные значения на тех наборах, на которых либо переменная x_1, либо переменная x_2 принимают единичные значения. Функция коммутативна. Для обозначения функции используют два функциональных символа.

$$\varphi_7 = x_1 \vee x_2 = x_1 + x_2$$

ИСКЛЮЧАЮЩЕЕ ИЛИ (сумма по модулю 2). Функция нечетности. Функция НЕ эквивалентность. Функция, которая принимает единичные значения на тех наборах, количество единиц, в которых нечетно. Функция коммутативна.

$\varphi_6 = x_1 \oplus x_2$

Функция *ЭКВИВАЛЕНТНОСТИ*. Функция четности. Функция, которая принимает единичные значения на тех наборах, количество единиц, в которых четно. Функция коммутативна.

$\varphi_9 = \overline{x_1 \oplus x_2} = x_1 \sim x_2$

Функция *ИМПЛИКАЦИЯ*. Функция, которая соответствует правилу: "из истины не может следовать ложь". Функция не коммутативна.

$\varphi_{11} = x_1 \to x_2$; $\varphi_{13} = x_2 \to x_1$

Функция *И-НЕ* (штрих Шеффера). Функция обратная функции логическое И.

$\varphi_{14} = \overline{x_1 \,\&\, x_2} = x_1 \mid x$

Функция *ИЛИ-НЕ* (стрелка Пирса). Функция обратная функции логическое ИЛИ

$\varphi_8 = \overline{x_1 \vee x_2} = x_1 \downarrow x$

На Рис.13.1 показано соответствие элементарных булевых функций элементарным логическим вентилям (элементарным логическим схемам).

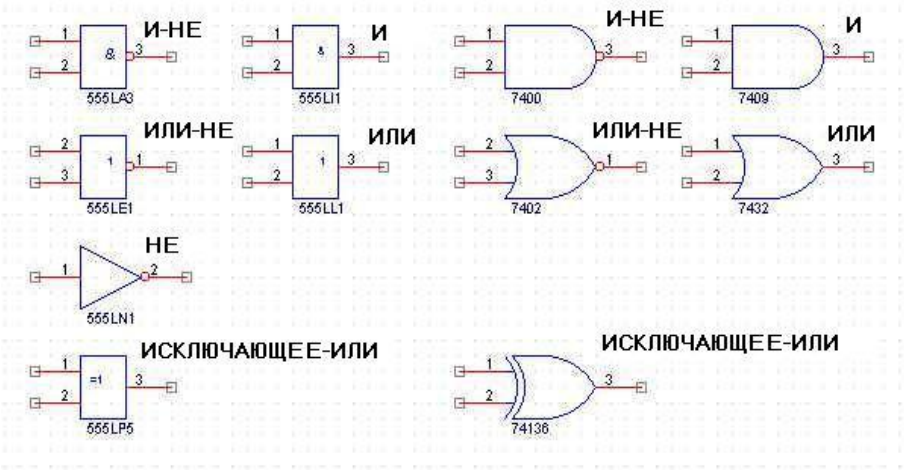


Рис.13.1

13.2. Эквивалентные формулы

Формулы называются эквивалентными, если на всяком наборе их переменных значения функций, реализуемых этими формулами, совпадают.

Определение: Любую схему, собранную из логических вентилей будем называть комбинационной схемой или комбинационным автоматом. В общем случае комбинационная схема может иметь *n* входных и *m* выходных полюсов (выводов или контактов). В большинстве случаев будем рассматривать схемы с *n* входными и 1 выходным полюсом. Любой такой схеме можно сопоставить булеву функцию от *n* переменных.

Рассмотрим отношения эквивалентности (Таблица 13.2), необходимые для эквивалентных преобразований (для упрощения формул).

Таблица 13.2

Коммутативность	$x \circ y = y \circ x$	$\&, \vee, \oplus, \sim, \mid, \downarrow$
Ассоциативность	$(x \circ y) \circ z = x \circ (y \circ z)$	$\&, \vee, \oplus, \sim$
Правило Де Моргана	$\overline{x \,\&\, y} = \bar{x} \vee \bar{y}$	$\overline{x \vee y} = \bar{x} \,\&\, \bar{y}$
Идемпотентность	$xx = x$	$x \vee x = x$
Двойное отрицание	$\bar{\bar{x}} = x$	
Свойства констант	$x \,\&\, 1 = x$	$x \,\&\, 0 = 0$
	$x \vee 1 = 1$	$x \vee 0 = x$
	$\bar{0} = 1$	$\bar{1} = 0$
Закон противоречия	$x\bar{x} = 0$	
Закон исключенного третьего	$\bar{x} \vee x = 1$	
Правила поглощения	$x \vee xy = x$	$x(1 \vee y) = x$
	$x \,\&\, (x \vee y) = x$	$x \vee xy = x$
	$x \vee \bar{x}y = x \vee y$	$x(\bar{x} \vee y) = xy$
Дистрибутивность	$x \vee (y \,\&\, z) = (x \vee y)(x \vee z)$	$x(y \vee z) = xy \vee xz$
	$x(y \oplus z) = xy \oplus xz$	
Склеивание	$xy \vee x\bar{y} = x$	
Расщепление	$x = xy \vee x\bar{y}$	
Обобщенное склеивание	$xz \vee y\bar{z} \vee xy = xz \vee y\bar{z}$	
Формулы	$x \oplus y = x\bar{y} \vee \bar{x}y$	$x \sim y = x \oplus y \oplus 1$
	$x \vee y = xy \oplus x \oplus y$	$x \to y = \bar{x} \vee y = xy \oplus x \oplus 1$

13.3. Специальные виды формул

Элементарная конъюнкция:

1. Пропозициональную переменную или её отрицание называем элементарной конъюнкцией.

2. Пусть ϕ_1 и ϕ_2 – элементарные конъюнкции, тогда $\phi_1 \wedge \phi_2$ тоже элементарная конъюнкция.

3. Других элементарных конъюнкций нет.

Дизъюнктивная нормальная форма (ДНФ):

1. Элементарная конъюнкция будет ДНФ.

2. Если ϕ_1 и ϕ_2 – ДНФ, то $\phi_1 \vee \phi_2$ – тоже ДНФ.

3. Других ДНФ нет.

Дизъюнктивная нормальная форма ϕ называется СДНФ, если

1. Любая пропозициональная переменная, входящая в ϕ, встречается в каждой элементарной конъюнкции, входящей в ϕ, ровно один раз;

2. Все элементарные конъюнкции, входящие в ϕ, различны.

Обычно *ДНФ* булевой функции строят по таблице истинности. Сначала по таблице получают *СДНФ*, а затем, применяя эквивалентные преобразования, упрощают ее, получая *ДНФ*.

СДНФ по таблице истинности построить достаточно просто. Для каждой строчки таблицы, которой соответствует единичное значение функции, записывают конъюнкцию ранга *n*. Если значение переменной в этой строчке было равно нулю, то в записываемую конъюнкцию данная переменная будет входить со знаком инверсии.

Пример 13.1: Рассмотрим булеву функцию f, заданную таблицей истинности:

x_3	x_2	x_1	y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

СДНФ данной функции будет равна: $f(X) = \bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1x_2x_3$

Пример 13.2: Дана схема комбинационного автомата (Рис.13.2). Необходимо написать формулу (ДНФ) функции, которую реализует данный автомат.

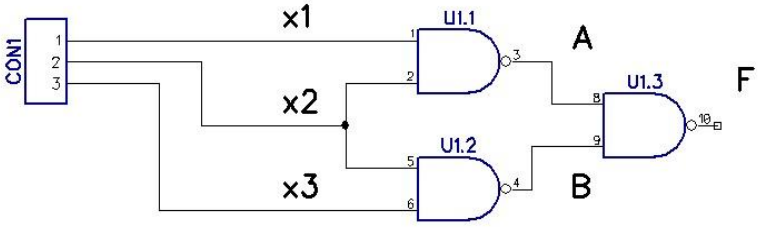


Рис.13.2

Обычно построение формулы по схеме начинают с конца (с последнего вентиля схемы) и записывают формулу, реализуемую последним элементом $F = \overline{AB}$. *Затем записывают формулы для полюсов (букв) A и B:* $A = \overline{x_1 x_2}$; $B = \overline{x_2 x_3}$; *Далее можно сразу подставить вместо A и B формулы, которые мы записали выше, а затем выполнять преобразования. Однако, можно несколько упр+остить формулы по отдельности, применив правило Де Моргана, а затем записать суперпозицию уже преобразованных формул.*

$F = \overline{A} \vee \overline{B}$, $A = \overline{x}_1 \vee \overline{x}_2$ $B = \overline{x}_2 \vee \overline{x}_3$ *После первого применения правила Де Моргана, соединения полученных функций в суперпозицию и нового применения правила Де Моргана получим следующую формулу:* $F = x_1 x_2 \vee x_2 x_3 = x_2(x_1 \vee x_3)$

14. История развития вычислительной техники

Прежде чем рассматривать современные устройства вычислительной техники, математические основы которых мы рассмотрели выше, приведем несколько исторических примеров.

Одним из первых устройств, которые люди стали использовать для вычислений, считается *абак* - (лат. *abacus* - *доска*) - счетная доска, применявшаяся для арифметических вычислений в Древней Греции и Древнем Риме приблизительно с V века д.н.э. В Западной Европе абак применялся до 18 в. Доска разделялась на полосы, счет осуществлялся передвижением находящихся в полосах счетных марок (костяшек, камней и т.п.). В странах Дальнего Востока распространен китайский аналог абака - *суан-пан*, в России - счеты.

Вот как рассказывает о своей встрече с абаком известный американский физик, лауреат Нобелевской премии, Ричард Фейнман.

Однажды в ресторан зашел японец, который продавал счеты, именуемые абаками. Он очень прилично владел мастерством вычислений на абаке и для привлечения внимания предлагал любому посоревноваться с ним в этом. Состязания со знаменитым физиком, который производил вычисления в уме, проходили в несколько этапов. В сложении чисел Фейнман проиграл. В умножении японец снова его опередил, зато в делении уже была ничья. Когда же японец предложил посоревноваться в вычислении кубического корня из числа 1729.03, то потерпел фиаско, так как ему было очень сложно вычислять знаки после запятой, а для Фейнмана выбранное число было просто удачным совпадением с известным ему близким результатом. Когда японец попросил физика объяснить как ему удается соревноваться на равных с абаком, то после нескольких попыток Фейнман понял, что его собеседник вообще ничего не смыслит в числах, не говоря уже о приближенных вычислениях. «Имея в руках абак, не нужно запоминать целую кучу арифметических комбинаций, довольно научиться передвигать вверх и вниз костяшки. Вы не обязаны помнить, что

9+7=16, вам достаточно помнить, что для прибавления 9 нужно сдвинуть десять костяшек вверх и одну вниз.» Эту фразу можно применить к любому вычислительному устройству, будь то простой механический калькулятор, или сложная электронная вычислительная машина. Любой алгоритм, реализуемый этими устройствами можно разложить на составляющие, представить в виде последовательности простых незамысловатых механических (или «электронных») действий, которые приводят к желаемому результату.

Если абак можно было бы назвать прототипом цифрового счетного устройства, то Антикитекрский механизм по праву мог бы считаться одним из первых аналоговых вычислительных устройств. Антикитерский механизм – механическое устройство, поднятое в 1901 году с древнего судна, затонувшего недалеко от греческого острова Антикитера и обнаруженного греческим водолазом 4 апреля 1900 года. Датируется приблизительно 100 годом до н. э. Механизм содержал 37 бронзовых шестерён в деревянном корпусе, на котором были размещены циферблаты со стрелками. До 1951 года артефакт оставался неисследованным, пока английский историк науки Джон Прайс не заинтересовался им. Прайс первый показал, что механизм является уникальным античным механическим вычислительным устройством. Исследования механизма, проведенные в конце 90-х годов XX века подтвердили гипотезу о том, что механизм служил для моделирования движения небесных тел и мог моделировать движение не только Солнца и Луны, но и пяти известных в древности планет – Меркурия, Венеры, Марса, Юпитера и Сатурна.

Далее рассмотрим еще несколько знаменитых имен, дат и конструкций, которые связаны с развитием вычислительной техники.

1492 год: Леонардо да Винчи в одном из своих дневников приводит эскиз 13-разрядного суммирующего устройства с десятизубцовыми кольцами. Хотя работающее устройство на базе этих чертежей было построено только в XX веке, всё же реальность проекта Леонардо да Винчи подтвердилась.

Вильгельм Шиккард (1592-1636), стал профессором восточных языков в университете г.Тюбингена в 1617 г.

Более 300 лет считалось, что автором первой счетной машины является Блез Паскаль. И вот в 1957 году директор Кеплеровского научного центра *Франц Гаммер* сделал в Германии на семинаре по истории математики сенсационный доклад, из которого следовало, что проект первой счетной машины появился как минимум на два десятилетия раньше «паскалева колеса», а сама машина была (по всей видимости) изготовлена в середине 1623 года.

Работая в городской библиотеке Штутгарта, Гаммер обнаружил фотокопию эскиза неизвестного ранее счетного устройства (оригинал находился в архиве выдающегося астронома и математика Иоганна Кеплера в Пулковской обсерватории близ Санкт-Петербурга). Гаммеру удалось установить, что этот эскиз есть не что иное, как отсутствовавшее приложение к

опубликованному ранее письму Иоганну Кеплеру профессора университета в Тюбингене Вильгельма Шиккарда (от 25 февраля 1624 года), где Шиккард, ссылаясь на чертеж, описывал изобретенную им счетную машину. Машина содержала суммирующее и множительное устройства, а также механизм для записи промежуточных результатов. Первый блок — шестиразрядная суммирующая машина — представлял собой соединение зубчатых передач. На каждой оси имелись шестерня с десятью зубцами и вспомогательное однозубое колесо — палец. Палец служил для того, чтобы передавать единицу в следующий разряд (поворачивать шестеренку на десятую часть полного оборота, после того как шестеренка предыдущего разряда сделает такой оборот). При вычитании шестеренки следовало вращать в обратную сторону. Контроль хода вычислений можно было вести при помощи специальных окошек, где появлялись цифры. Для перемножения использовалось устройство, чью главную часть составляли шесть осей с “навернутыми” на них таблицами умножения.

После доклада Гаммера исследователи также вспомнили и про другое письмо Шиккарда Кеплеру — от 20 сентября 1623 года, — на которое прежде не обращали должного внимания. В нем Шиккард сообщал, что осуществил механически то, что Кеплер делал алгебраически, а именно — сконструировал машину, автоматически выполняющую сложение, вычитание, умножение и деление. Шиккард писал, что Кеплер был бы приятно удивлен, если бы увидел, как машина сама накапливает и переносит влево десяток или сотню и как она отнимает то, что держит в «уме» при вычитании.

Джон Непер (*англ. John Napier; 1550-1617*) - шотландский барон, математик. В работе "Описание удивительной таблицы логарифмов" (1614) изложил свойства логарифмов, дал описание таблиц, правила пользования ими и примеры применений. Основанием таблицы логарифмов Непера является иррациональное число, к которому неограниченно приближаются числа вида $(1 + 1/n)^n$ при безграничном возрастании n. Это число называют *неперовым* числом и обозначают буквой e. Открытые Непером логарифмы и логарифмические таблицы позволили заменять умножение и деление соответственно сложением и вычитанием. Работа Непера *«Канон о логарифмах»* начинался так: «Осознав, что в математике нет ничего более скучного и утомительного, чем умножение, деление, извлечение квадратных и кубических корней, и что названные операции являются бесполезной тратой времени и неиссякаемым источником неуловимых ошибок, я решил найти простое и надежное средство, чтобы избавиться от них». Можно привести еще одну цитату из работ Непера: «Отбросьте числа, произведение, частное или корень которых необходимо найти, и возьмите вместо них такие, которые дадут тот же результат после сложения, вычитания и деления на два и на три». Иначе говоря, используя логарифмы, операцию умножения можно упростить до сложения, деление заменить вычитанием, а извлечение квадратного и кубического корней – в деление на два и на три соответственно. Например, нам

надо умножить число 3,8 на число 6,61. Для получения результата необходимо определить с помощью таблицы логарифмы множителей и затем сложить их: 0,58+0,82=1,4. На последнем шаге в таблице логарифмов находят число, логарифм которого равен полученной сумме, и получим почти точное значение искомого произведения: 25,12.

1630 год: Ричард Деламейн создаёт круговую логарифмическую линейку. Уильям Отред (*англ. William Oughtred, 5 марта 1575-30 июня 1660*) – английский математик, внёс решающий вклад в изобретение логарифмической линейки. Он предложил использовать две одинаковые шкалы, скользящие одна вдоль другой. Саму идею логарифмической шкалы предложил *Э. Гантер,* но для выполнения вычислений шкалу *Гантера* нужно было измерять двумя циркулями что влияло на удобство и точность вычислений. Двойная шкала *Отреда* сразу давала результат. В 1662 году *Сет Партридж* изобрёл бегунок и визир, и в этом виде логарифмическая линейка дожила до наших дней и служила инженерам и математикам до середины двадцатого века более трехсот лет.

В основе вычислений с использованием логарифмической линейки лежат следующие соотношения:

$log(x) + log(y)=log(xy)$

$log(x) - log(y)=log(x/y)$

$x+y=(x/y+1)*y$

$x-y=(x/y - 1)*y$

Для того чтобы вычислить произведение двух чисел, начало подвижной шкалы совмещают с первым множителем на неподвижной шкале, а на подвижной шкале находят второй множитель. Напротив него на неподвижной шкале находится результат умножения этих чисел Чтобы разделить числа, на подвижной шкале находят делитель и совмещают его с делимым на неподвижной шкале. Начало подвижной шкалы указывает на результат. С помощью логарифмической линейки находят лишь мантиссу числа, его порядок вычисляют в уме. Точность вычисления обычных линеек — два-три десятичных знака.

Блез Паска́ль (фр. Blaise Pascal ; 19 июня 1623,—19 августа 1662), начал создавать суммирующую машину «Паскалину» в 1642 году в возрасте 19 лет, наблюдая за работой своего отца, который был сборщиком налогов и часто выполнял долгие и утомительные расчёты. Машина Паскаля представляла собой механическое устройство в виде ящичка с многочисленными связанными одна с другой шестерёнками. Принцип работы машины Паскаля основывался на счёте оборотов колесиков. Складываемые числа вводились в машину при помощи соответствующего поворота наборных колёсиков. На каждое из этих колёсиков, соответствовавших одному десятичному разряду числа, были нанесены деления от 0 до 9. При вводе числа, колесики прокручивались до

соответствующей цифры. Совершив полный оборот, избыток над цифрой 9 колёсико переносило на соседний разряд, сдвигая соседнее колесо на 1 позицию. Первые варианты «*Паскалины*» имели пять зубчатых колёс, позднее их число увеличилось до шести или даже восьми, что позволяло работать с большими числами, вплоть до 9999999. Ответ появлялся в верхней части металлического корпуса. Вращение колёс было возможно лишь в одном направлении, исключая возможность непосредственного оперирования отрицательными числами. Тем не менее, машина Паскаля позволяла выполнять не только сложение, но и другие операции, но требовала при этом применения довольно неудобной процедуры повторных сложений. Вычитание выполнялось при помощи дополнений до девятки, которые для помощи считавшему появлялись в окошке, размещённом над выставленным оригинальным значением.

Так как успех в осуществлении замысла зависел от того, насколько точно ремесленники воспроизводили размеры и пропорции деталей машины, Паскаль сам присутствовал при изготовлении её составляющих. После попытки подделки его машины одним часовщиком Паскаль временно прекратил работу над своим изобретением. Чтобы побудить его продолжить совершенствование машины, друзья привлекли к ней внимание канцлера *Сегье*. Тот, изучив проект, рекомендовал Паскалю не останавливаться на достигнутом. В 1645 году Паскаль преподнёс Сегье готовую модель машины. До 1652 года под его наблюдением было создано около 50 вариантов «паскалины». В 1649 году он получил королевскую привилегию на счётную машину: возбранялись как копирование модели Паскаля, так и создание без его разрешения любых других видов суммирующих машин; запрещалась их продажа иностранцами в пределах Франции. Изобретённый Паскалем принцип связанных колёс почти на три столетия стал основой создания большинства арифмометров.

Го́тфрид Ви́льгельм Ле́йбниц (нем.Gottfried Wilhelm Leibniz 21 июня [1 июля] 1646 — 14 ноября 1716) - немецкий философ, логик, математик, физик, юрист, историк, дипломат, изобретатель и языковед. Основатель и первый президент Берлинской Академии наук, иностранный член Французской Академии наук

Лейбниц изобрёл собственную конструкцию арифмометра, гораздо лучше паскалевской, — он умел выполнять умножение, деление, извлечение квадратных и кубических корней, а также возведение в степень. Предложенные Готфридом ступенчатый валик и подвижная каретка легли в основу всех последующих арифмометров вплоть до XX столетия.

Современная двоичная система была полностью описана Лейбницем в работе *Explication de l'Arithmétique Binaire.* В системе счисления Лейбница были использованы цифры 0 и 1, как и в современной двоичной системе. Как человек, увлекающийся китайской культурой, Лейбниц знал о книге Перемен и заметил, что *гексаграммы* соответствуют двоичным числам от 0 до 111111. Он

восхищался тем, что это отображение является свидетельством крупных китайских достижений в философской математике того времени

В 1673 году Лейбниц в Лондоне на заседании Королевского общества продемонстрировал свой арифмометр, и его избрали членом Общества

Христиан Людвиг Герстен (7 февраля 1701 - 13 августа 1762) Член Лондонского королевского общества немецкий математик, физик, астроном в 1723 году изобрел арифметическую машину, а два года спустя ее изготовил. Машина Герстена создана на основе работ Лейбница, и замечательна тем, что в ней впервые применено устройство для подсчета частного и числа последовательных операций сложения, необходимых при умножении чисел. Также был предусмотрен контроль за правильностью установки входных данных.

Иоганн Мюллер (Johann Helfrich Müller -16 января 1746 - 12 июня 1830) немецкий военный инженер. в книге, изданной в 1788 году. выдвигает идею «разностной машины» — специализированного калькулятора для табулирования логарифмов, вычисляемых разностным методом. Считается, что Иоганн Мюллер был первым, кто предложил использовать метод разностей при вычислениях и сделать вычислительную машину, печатающую результаты.

Механизм универсального калькулятора И. Мюллера был основан на знаменитых ступенчатых валиках Готфрида В. Лейбница. Эта машина (13 см в высоту и 30 см в диаметре) могла выполнять все четыре действия арифметики над 14-разрядными числами. Если пользователь машины делал какие-то некорректные установки на машине, то звонок предупреждал его об этом. В настоящее время универсальный калькулятор Иоганна Мюллера хранится в Германии, в Hessisches Landesmuseum в Дармштадте. В 1784 г. в своем письме Мюллер говорит, что его машина будет вычислять и печатать арифметическую прогрессию. Это письмо содержит описание самого первого печатающего калькулятора. В другом письме, написанном в тот же год, Мюллер высказывает мысль о создании машины для вычисления площадей и объемов при помощи разностей. Мюллер придумал свою разностную машину за 36 лет до великого английского математика Ч. Бэббиджа.

Жозе́ф Мари́ Жаккáр (иногда Жаккард; фр. Joseph Marie Jacquard; 7 июля 1752, Лион — 7 августа 1834, Уллен, Рона) — французский изобретатель ткацкого станка для узорчатых материй (*машина Жаккарда*). Яркий пример машины с программным управлением, созданной задолго до появления вычислительных машин. Для того, чтобы задать необходимый узор использовалась набранная двоичным кодом перфокарта. Кодирование – есть отверстие – нет отверстия. Соответственно, какая-то нить поднялась, какая-то нет. Челнок прокидывает в образовавшийся зев нить, формируя двусторонний орнамент, где одна сторона является цветовым или фактурным негативом другой. Поскольку для создания даже некрупного узора, требуется около 100 и более уточных нитей и ещё большее количество нитей основы, создавалось

огромное количество перфорированных карт, которые связывались в единую ленту, Прокручиваясь, она могла занимать два этажа. Одной перфокарте соответствует один прокид челнока.

Тома де Кольмар Шарль Ксавье (Charles Xavier Thomas de Colmar) (1785, Кольмар, Эльзас — 1870, Париж) — французский предприниматель, создатель первого коммерческого арифмометра.

В 1820 году Тома создал арифмометр, основанный на принципе калькулятора Лейбница, способный производить умножение и деление. По своим возможностям арифмометр превосходил все известные в то время машины, так как мог оперировать тридцатизначными числами. Хотя Тома получил патент на свой арифмометр в том же году, первые арифмометры появились в продаже лишь в начале 1840-х годов.

Своим широким распространением во второй половине 19 века арифмометр обязан отнюдь не прогрессу в области вычислительной техники, а конкуренции на рынке вычислительных средств. Сам Тома прикладывал огромные усилия для популяризации своего детища. В конце концов этот надежный прибор прочно занял свое место на конторских столах и успешно продавался в течение последующих 90 лет.

В 1844 году Тома представил свой арифмометр на французской национальной выставке промышленных товаров, который, однако, уступил пальму первенства счетной машине Рота. В 1849 году Тома решил вновь попытать счастья. На этот раз он был удостоен серебряной медали и трехстраничного отчета жюри, но высшую награду получил «Аритморель» Мореля и Жейе, вычислительная машина, свойства которой были оценены выше. В 1851 году в Лондоне Тома постигло новое разочарование: его арифмометр уступил первенство калькулятору Стаффела.

После этой череды неудач Тома еще активнее принялся рекламировать свой арифмометр. В эти годы он интенсивно переделывал и улучшал свой прибор. Тома опубликовал описание арифмометра и разослал его всем коронованным и знатным особам Европы, благодаря чему удостоился многих наград в период с 1851 по 1855 годы. К Всемирной выставке в Париже 1855 года Тома построил гигантский двухметровый арифмометр, который занимал целый стол и обслуживать его должны были сразу два человека. Однако удача вновь изменила ему — золотую медаль получил дифференциальный анализатор шведских инженеров Георга и Эдварда Шойтцев. По иронии судьбы шведской машине была уготована короткая жизнь — было продано всего лишь два ее экземпляра, в то время как арифмометр постепенно добивался признания. С конца 1870-х, когда выпуск арифмометров значительно вырос, за ними закрепилась репутация первого стандарта в вычислительной технике. За свое изобретение Тома был награжден орденом Почетного легиона.

Ча́рльз Бэ́ббидж (англ. Charles Babbage; 26 декабря 1791, Лондон, Англия — 18 октября 1871, там же) — английский математик, изобретатель первой аналитической вычислительной машины. Иностранный член-корреспондент Императорской академии наук в Санкт-Петербурге (1832). Труды по теории функций, механизации счёта в экономике. Сконструировал и построил (1820-22) машину для табулирования. С 1822 работал над постройкой разностной машины. В 1833 разработал проект универсальной цифровой вычислительной машины — прообраза современной ЭВМ.

В 1819 году, когда Бэббидж заинтересовался астрономией, он более точно определил свои идеи и сформулировал принципы вычисления таблиц разностным методом при помощи машины, которую он впоследствии назвал разностной. Эта машина должна была производить комплекс вычислений, используя только операцию сложения. В 1819 году Чарльз Бэббидж приступил к созданию малой разностной машины, а в 1822 году он закончил её строительство и выступил перед Королевским Астрономическим обществом с докладом о применении машинного механизма для вычисления астрономических и математических таблиц. Он продемонстрировал работу машины на примере вычисления членов последовательности. Работа разностной машины была основана на методе конечных разностей. Малая машина была полностью механической и состояла из множества шестерёнок и рычагов. В ней использовалась десятичная система счисления. Она оперировала 18-разрядными числами с точностью до восьмого знака после запятой и обеспечивала скорость вычислений 12 членов последовательности в 1 минуту. Малая разностная машина могла считать значения многочленов 7-й степени.

За создание разностной машины Бэббидж был награждён первой золотой медалью Астрономического общества. Однако малая разностная машина была экспериментальной, так как имела небольшую память и не могла быть использована для больших вычислений.

В 1822 году Бэббидж задумался о создании большой разностной машины, которая позволила бы заменить огромное количество людей, занимающихся вычислением различных астрономических, навигационных и математических таблиц. В 1823 Королевское и астрономическое общество выделили Бэббиджу 1500 фунтов стерлингов и он приступил к разработке новой машины. Он планировал сконструировать машину за 3 года. Однако Бэббидж не учёл сложности конструкции, а также технические возможности того времени. К 1827 году было затрачено 3500 фунтов стерлингов (более 1000 личных денег). В 1827 году. он похоронил отца, жену и двоих детей. После этих событий у него ухудшилось самочувствие, и он не мог заниматься конструированием машины. Чтобы восстановить здоровье, он поехал в путешествие по континенту.

После путешествия в 1828 году Бэббидж продолжил разработку, но денег уже не было. Он обращался ко многим обществам и правительству с просьбой о помощи. Только в 1830 году он получил от правительства ещё 9000 фунтов стерлингов, после чего продолжил конструирование разностной машины.

К 1834 году на создание машины уже было потрачено 17000 фунтов государственных денег и 6000 личных и работы опять были приостановлены. С 1834 по 1842 год правительство рассматривало вопрос об оказании поддержки проекта, а в 1842 году было принято решение об отказе. Разностная машина так и не была достроена.

Большая разностная машина должна была состоять из 25 000 деталей, весить почти 14 тонн и быть 2,5 метра высотой. Кроме того, разностная машина должна была быть оснащена печатным устройством для вывода результатов. Память была рассчитана на 1000 50-разрядных чисел.

Возможно, одной из причин неудач с разностной машиной стала излишняя разносторонность Бэббиджа. Он поднимался с экспедицией на Везувий, погружался на дно озера в водолазном колоколе, участвовал в археологических раскопках, изучал залегание руд, спускаясь в шахты. Почти год он занимался безопасностью железнодорожного движения и сделал очень много специального оборудования — в том числе создал спидометр. Кроме того, при конструировании разностной машины он разработал немало оборудования для обработки металла. В 1851 году Чарльз Бэббидж предпринял попытку сконструировать улучшенную версию разностной машины — «Разностную машину 2». Но и этот проект не был удачным.

Несмотря на неудачу с разностной машиной, Бэббидж в 1834 году задумался о создании программируемой вычислительной машины, которую он назвал аналитической (прообраз современного компьютера). В отличие от разностной машины, аналитическая машина позволяла решать более широкий ряд задач. Именно эта машина стала делом его жизни и принесла посмертную славу. В единую логическую схему Бэббидж увязал арифметическое устройство (названное им «мельницей»), регистры памяти, объединенные в единое целое («склад»), и устройство ввода/вывода, реализованное с помощью перфокарт трех типов. Перфокарты операций переключали машину между режимами сложения, вычитания, деления и умножения. Перфокарты переменных управляли передачей информации со «склада» на «мельницу» и обратно. Числовые перфокарты могли быть использованы как для ввода данных в машину, так и для сохранения результатов вычислений, если место на «складе» было ограничено

Бэббидж часто посещал промышленные выставки, где были представлены различные новинки науки и техники. Именно там состоялось его знакомство с Адой Августой Лавлейс (дочерью Джорджа Байрона), которая стала его очень близким другом, помощником и единомышленником. В 1840 году Бэббидж ездил по приглашению итальянских математиков в Турин, где читал лекции о

своей машине. Итальянцы подготовили и опубликовали конспект лекций на французском языке. Позже Ада Лавлейс перевела эти лекции на английский язык, дополнив их комментариями по объёму превосходящими исходный текст. В комментариях Ада сделала описание ЦВМ и инструкции по программированию к ней. Это были первые в мире программы. Именно поэтому Аду Лавлейс справедливо называют первым программистом. Однако, аналитическая машина так и не была закончена.

Георг Шутц (Георг Шойц) (швед. Per Georg Scheutz; 23 сентября 1785 г. — 22 мая 1873 г.) — шведский изобретатель, наиболее известен по его новаторским работам в компьютерных технологиях. Изобретённая в 1837 г. и собранная в 1843 г. машина, которую он сделал со своим сыном *Эдвардом Шутцем,* была основана на разностной машине Чарльза Бэббиджа. Улучшенная модель, примерно в размер фортепиано, была создана в 1853 г. и впоследствии демонстрировалась на Всемирной выставке в Париже в 1855 г. В 1859 г. машина была продана британскому правительству. *Шутц* создал ещё одну машину в 1860 г. и продал её Соединенным Штатам. Машина предназначалась для создания логарифмических таблиц.

Машина была несовершенной и не могла создавать полные таблицы, Мартин Виберг переделал конструкцию Шутца и в 1875 г. создал компактное устройство, которое печатало полные таблицы.

Герман Холлерит (англ. Herman Hollerith, иногда по-русски применяется написание Голлерит; 29 февраля 1860 — 17 ноября 1929) — американский инженер и изобретатель немецкого происхождения

В 1880 году Биллингс - авторитетный эксперт по анализу статистических данных - сообщил Холлериту о своей идее создания машины с перфокартами для составления таблиц по данным переписи населения США. идея состояла в том, что различные статистические данные можно записывать на одной карточке путем перфорирования, а затем обрабатывать с помощью механических средств, выбирая необходимые группы перфораций.

В 1882 году, Холлерит работает в Массачусетском технологическом институте в качестве преподавателя по машиностроению. Он провел там год, одновременно развивая свои идеи и разрабатывая первое оборудование для записи данных переписи и составления по ним таблиц. В 1883 году Холлерит вернулся в Вашингтон, где работал в патентном бюро. Знание патентных систем помогло ему в следующем десятилетии как изобретателю. В 1884 году в Сант-Луисе он разработал идею об улучшении тормозов для железнодорожного транспорта. В то время он уже мог построить прототип табулятора, но у него не было на это денег. Известно, что ему не удавалось занять деньги у своей семьи или друзей на нужды своих проектов, поэтому он построил прототип на свои скудные сбережения.

Патент 395.782 «Искусство составления статистических данных» стал самым значимым патентом Холлерита. Он был зарегистрирован 23 сентября 1884 года. Благодаря своему опыту, Холлерит подтвердил, что перфокарты являются важнейшей частью в процессе составления таблиц. Многие промышленники должны были заключить договор с Холлеритом о лицензии на его оборудование с перфокартами. Следующий патент 397.783 был назван «Устройство для составления статистических данных».

Во время переписи населения 1890 года данные о каждом человеке переносили на карты размером 73/8×33/4 дюйма. Затем делали перфорацию по краям, согласно каждой характеристике. Один угол карты подрезался по диагонали для удобства при подсчете и сортировке (выборке).

Только лишь спустя 14 лет Холлерит изобрел полностью автоматическую систему сортировки.

Холлерит получил широкую известность за свои работы, но в 1890 году его успех оказался абсолютно непредвиденным, когда он заключил контракт на 11 переписей населения после победы в соревновании по переписи населения в четырех районах Сант-Луиса, имеющего более 10 491 жителей.

Метод Холлерита был не только самым быстрым, но и самым точным. Было подсчитано, что Холлерит сэкономил государству 597 125 долларов. Во время переписи населения вновь было подсчитано, что он сэкономил два года и большую сумму денег.

В 1890 году Холлериту исполнилось 30 лет. Он получил звание доктора философии, был всемирно известен за свое изобретение табулятора и перфокарт, а также заключил очень важный контракт с Бюро переписи населения США. Казалось, что его финансовое и профессиональное будущее обеспечено. Вначале Холлерит руководил фирмой «Электрическая система составления таблиц Холлерита» и мастерской в Вашингтоне, где собирали и ремонтировали вычислительные машины, а также делали перфокарты. во избежание неверных слухов о плохом качестве машин Холлерит ремонтировал или заменял вышедшие из строя части. Таким образом в торговом соглашении появился пункт о сервисном обслуживании машин.

К 1895 году машины Холлерита работали уже в Австрии, в Канаде, шли переговоры об их продаже Италии и России. Позже его вычислительные машины использовались и в других зарубежных странах

3 декабря 1896 года Холлерит зарегистрировал фирму *Tabulating Machine Company* с уставным капиталом в 100 000 USD, разделенных на 1000 долей по 100 USD каждая Tabulating Machine Company самостоятельно не занималась производством табулирующих машин. Она размещала заказы у хорошо зарекомендовавших фирм, таких как *Pratt & Whitney u Western Electric*. С 1901 г. Холлерит был собственником компании Taft-Peirce, производителя своих машин, но в 1905 г. продал ее.

В 1896 году Холлерит добрался до России. 15 декабря 1896 г. был подписан контакт об обеспечении Русского Правительства электрическими табулирующими машинами, изобретенными Холлеритом. Фирма Холлерита предоставила России в аренду 35 старых машин, а также продала 500 перфораторов и 70 табуляторов с сортировальными машинами за $67571. Их сборка осуществлялась в С.-Петербурге Перепись в России можно охарактеризовать следующими данными: 129 млн. граждан (129 211 113), 900 тыс. счетчиков и 2 тыс. аналитиков в 14 категориях данных. Холлерит неоднократно приезжал в Россию для консультаций. Надо отметить, что первая всеобщая перепись в России не может служить образцом применения машин Холлерита. Ее результаты были опубликованы только в конце 1905 года. Эти причины в недостаточной квалификация персонала, отсутствие помощи от местных властей, недостаточной формализации записей. Скажем, император Николай II написал в графе профессия “Хозяин Земли Русской” – и как пробить это на перфокарте, а потом и сортировать.

Однако, недовольное финансовыми требованиями Холлерита Бюро переписи начало производить свои вычислительные машины, полагаясь в основном на прошлых служащих Холлерита, которые раскрывали им ноу-хау. Бюро заимствовало идеи Холлерита и заменило некоторые механические части на электрические механизмы.

Опасения утраты контракта с Бюро переписи вынудили Холлерита искать новые рынки сбыта своих систем в начале 1900-х годов. К 1910 году в списке его клиентов значились многочисленные страховые компании, железные дороги, универмаги, заводы, энергетические компании и др.

Несмотря на уплывшие выгодные госзаказы оборот *Tabulating Machine Company* в 1910 году достиг 350 тыс. USD и показывал постоянный рост. Тем не менее Холлерит в 1911 году в ответ на предложение всемирно известного предпринимателя *Чарльза Флинта*, образовавшего многие промышленные империи принял решение об объединении (а фактически о продаже) с двумя другими компаниями: *International Time Recording Company (ITR*) (занималась производством табельных часов) и *Computing Scale Company of America* (образовалась из четырех компаний, производящих весы и машинки для нарезания продуктов). Объединение стало называться *Computing-Tabulating-Recording, или C-T-R*. Сам Г. Холлерит продолжал работу в объединенной компании в качестве инженера-консультанта до 1921 года. В 1924 г. C-T-R стала называться IBM.

Говоря о вычислительной технике, нельзя обойти стороной аналоговые вычислительные машины.

1927 год: в Массачусетском технологическом институте (MIT) был изобретен аналоговый компьютер. Аналоговые устройства оказались востребованы по причине простоты замещения (моделирования) аналоговыми

схемами дифференциальных уравнений. Вот ретроспектива событий, связанных с созданием аналогового компьютера

- *1878 год*, польский математик *Абданк-Абаканович* разработал теорию интерграфа – некоего аналогового интегрирующего устройства, позволяющего получить интеграл от произвольной функции, изображённой на плоском графике.

- *1890 год* – начато серийное производство арифмометров *Однера* – самого распространённого типа арифмометров XX века. К арифмометрам Однера относится, в частности, знаменитый «Феликс».

- *1904 год*, российский инженер *А.Н.Крылов* изобрел первую механическую вычислительную машину, решающую дифференциальные уравнения (применялась при проектировании кораблей).

- *1912 год* – создана машина для интегрирования обыкновенных дифференциальных уравнений по проекту российского учёного А.Н.Крылова.

- *1919 год* – Появился *Mercedes-Euklid VII* – первый в мире вычислительный автомат, то есть арифмометр, способный, самостоятельно осуществлять все четыре основных арифметических действия.

- *1930 год – Ванневар Буш* (США) создал механическую интегрирующую машину, применяющийся при расчёте траектории стрельбы корабельных орудий. (в 1942 году – создана её электромеханическая версия).

- *1935 год,* СССР – выпуск первой советской электродинамической счётно-аналитической машины *САМ* (модель Т-1). Разработаны механический интегратор и электрический расчётный стол для определения стационарных режимов энергетических систем.

- *1942–1944 годы*, США –операционный усилитель постоянного тока, имеющий достаточно высокий коэффициент усиления, что дало возможность конструировать аналоговые компьютеры без движущихся частей, на постоянном токе.

- *1945 –1946* годы, СССР – под руководством Л. И. Гутенмахера изобретены первые электронные аналоговые машины с повторением решения.

- *1949 год*, СССР – изобретён ряд АВМ на постоянном токе, что положило начало развитию аналоговой вычислительной техники в СССР.

- *1960-е* годы, аналоговые компьютеры являлись повседневным инструментом ученых для решения множества специфических задач в различных областях науки. В СССР расцвет электронных аналоговых вычислительных машин с их серийным выпуском пришёлся на 1960–1970-е годы.

Говоря о цифровых вычислительных машинах, нельзя не упомянуть о немецком инженере – изобретателе *целой линейки вычислительных машин Конраде Цузе.*

1938 год: немецкий инженер *Конрад Цузе* вскоре после окончания в 1935 году Берлинского политехнического института изобрел свою первую полностью механическую программируемую цифровую машину, названную Z1. В качестве его соавтора упоминается также Хельмут Шрейер (Helmut Schreyer). Модель была пробной и в практических целях не использовалась. Ее восстановленная версия хранится в Немецком техническом музее в Берлине. В том же году Цузе приступил к созданию машины Z2.

1941 год: *Конрад Цузе* создаёт первую вычислительную машину Z3, обладающую всеми свойствами современного компьютера.

В *1944* году *Конрад Цузе* разработал ещё более быстродействующий компьютер Z4.

В сороковые годы XX столетия работы над цифровыми компьютерами велись и в Соединенных Штатах и в Англии.

1942 год: в Университете штата *Айова* успешно протестировано первое в США цифровое вычислительное устройство (первая вычислительная машина без движущихся частей) – компьютер *ABC (Atanasoff-Berry Computer*). Компьютер *ABC* был задуман *Джоном Атанасовым* и его аспирантом *Клиффордом Берри* в *1937* году. Работы над проектом начались в 1939 году, а закончились созданием опытного образца в *1942* году. Компьютер *ABC* содержал 1,5 тысячи электронных ламп. Хотя эта машина так и не была завершена (Атанасов ушел в действующую армию), она, как пишут историки, оказала большое влияние на *Джона Моучли*, создавшего двумя годами позже ЭВМ *Эниак*.

В начале *1943* года в Гарвардском университете успешные испытания прошла электромеханическая вычислительная машина *Mark I*, или *ASCC (Automatic Sequence Controlled Calculator*), предназначенная для выполнения сложных баллистических расчётов американского ВМФ. Машина была около 17 метров в длину, содержала 75 000 электронных ламп и 3 000 механических реле. *МАРК I* производил вычисления с точностью до 23-го значащего знака и выполнял операцию сложения за 3 секунды, а деления - за 12 секунд. Таким образом, у данного компьютера показатель вычислительной мощности был меньше единицы!

В конце *1943* года заработала английская вычислительная машина специального назначения *Колосс.* Машина работала над расшифровкой секретных кодов фашистской Германии.

1946 год стал годом создания первой универсальной электронной цифровой вычислительной машины Эниак – *ENIAC* (*Electronic Numerical*

Integrator and Computer – электронный числовой интегратор и вычислитель). Авторы проекта *Джон Моучли* и *Преспер Эккерт*.

Считается, что первая удачная передача данных между двумя компьютерами произошла *9 сентября 1940* года, когда *Джордж Стибиц,* сотрудник *Bell Labs*, при помощи телетайпа передал два числа и команду, предписывающую разделить одно число на другое. Адресатом и исполнителем было электромеханическое вычислительное устройство, которое находилось на расстоянии нескольких сотен километров от заказчика вычисления. Ответ успешно пришел по кабелю; все вычисление заняло тридцать с небольшим секунд.

Основу для построения систем внешней памяти – технологию цифровой магнитной записи разработала еще в *1945* году фирма *IBM*. Фирма *UNIVAC* на основе этого изобретения выпустила первую цифровую память на магнитной ленте. В *1953* году *Джеймс Форрест* изобрел ферритовую память в виде кольцевых сердечников и тогда же появился первый серийно производимый накопитель на магнитной ленте - устройство IBM 726, плотность записи которого составляла 100 символов на дюйм, скорость - 75 дюймов в секунду. В 1966 году IBM применила первую подсистему дисковой памяти - *IBM RAMAC* 305. Она имела емкость 5 Мбайт на 50 двухфутовых пластинах. Год спустя, в 1967 году, *А.Бобек,* сотрудник Bell Labs, разработал память на цилиндрических доменах (*bubble memory*). Первый накопитель на гибких магнитных дисках появился в *1971* году в мэйнфреймах фирмы IBM Sistem 370/135 и 370/195.

Исследованием полупроводников занимались многие ученые, однако наиболее известны эксперименты *Уильяма Бедфорда Шокли 1947* года. Именно эта дата фигурирует в большинстве источников как дата изобретения транзистора. В *1956* году за труды в области полупроводниковой техники *Бедфорду Шокли* была присуждена Нобелевская премия. Его открытие определило сущность второго поколения компьютеров - компьютеров на базе полупроводниковых элементов. Полупроводниковые технологии позволили не только повысить надежность, но и уменьшить габариты машин. В начале 60-х компания DEC разработала свой первый миникомпьютер *PDP-1*, а через два года начались продажи компьютера *PDP-5*.

В *1964* году была разработана первая локальная сеть в США, в Ливерморской лаборатории. В 1968 году под эгидой Агенства по перспективным исследованиям министерства обороны США (ARPA) началась разработка и внедрение глобальной военной компьютерной сети, связывающей исследовательские лаборатории на территории США. В 1971 году сеть ARPANet уже насчитывала 15 узлов. В 1984 году Национальный научный фонд США разработал глобальную сеть NSFnet, которая соединяла суперкомпьютеры, расположенные в различных университетах и исследовательских центрах. Между NSF и военными сетями создавались мосты и было начато использование стека протоколов TCP/IP. В 1984 году при

непосредственном участии *Тома Дженнингса и Джона Мэдила* появилась некоммерческая компьютерная сеть FIDO.

В *1969* году сотрудник *Intel Тед Хофф* изобрел микропроцессор. В 1970 году другой сотрудник Intel *Фредерико Фагин* начал работы по проектированию микропроцессора. А через год появился первый в мире четырехразрядный микропроцессор *Intel 4004*, содержащий 2300 транзисторов на кристалле, его тактовая частота составляла 108 кГц, быстродействие 60 000 операций в секунду, адресуемая память 640 байт, цена 200 долларов.

В 1974 году на базе процессора Intel 8080 (первого восьмиразрядного процессора, появившегося в этом же году) был спроектирован компьютер «Альтаир 8800», который некоторые эксперты называют первым персональным компьютером в истории развития техники.

В Советском Союзе первая электронная вычислительная машина была создана в Киеве группой *С.А. Лебедева в 1950* году.

В пятидесятые и шестидесятые годы в нашей стране были получены также неплохие результаты при создании ЭВМ первого и второго поколений. Разными авторскими коллективами был разработан целый ряд оригинальных вычислительных машин и комплексов, которые с успехом использовались как в народном хозяйстве, так и при создании оборонных систем. Работы выполнялись под руководством таких выдающихся конструкторов, как *С.А. Лебедев* (мэйнфреймы «*МЭСМ*», «*БЭСМ-1*» — «БЭСМ-6»), *В.М. Глушков* (серия «МИР1» — «МИР3»), *Б.И. Рамеев* (семейства «Урал» и «Стрела»), *И.С. Брук* (малые и управляющие машины), *М.А. Карцев* (многомашинные вычислительные комплексы и кластеры*), Н.Я. Матюхин* (вычислительные машины для системы ПВО), *Г.П. Лопато* («М-3» и семейство «Минск»). К сожалению, при таком многообразии вычислительных машин и комплексов возникла и не была решена очень серьезная проблема — проблема программной совместимости как между различными семействами ЭВМ, так и между поколениями внутри одного семейства. Талантливые разработчики ЭВМ и программного обеспечения для них работали разобщенно, вне рамок какой-либо государственной программы и требований стандартизации. Это привело сначала к потере темпа, потом к отставанию и, наконец, к роковому для отечественной информатики копированию в СССР и странах Восточной Европы разработок фирмы IBM. После этого наши достижения в области программирования, операционных систем, языков программирования практически сошли на нет, что и отмечается зарубежными экспертами.

Отдельной строкой можно отметить работы по параллельному программированию. Начало исследований в этой области относится к середине 60-х годов, когда в институте математики СО АН СССР (Новосибирск) и в московском энергетическом институте возникли первые коллективы, заинтересовавшиеся теорией параллельных процессов в вычислительных системах, состоящих из однородных или неоднородных машин. Отечественные

специалисты первыми в мировой науке дали постановку и предложили первые решения таких задач, как сегментация алгоритмов и программ, планирование выполнения больших программ на вычислительных системах, динамическое диспетчерирование потока программ и сегментов программ, асинхронная организация протекания процессов. В это время было предложено несколько оригинальных моделей для параллельных вычислений, заново переоткрытых потом в США и других странах.

Также заслуживает упоминания школа *М.Л. Цетлина — М.М. Бонгарда*. Вокруг этих двух ярких и самобытных людей объединились математики, программисты, физиологи, инженеры и физики. Основной круг интересов участников школы концентрировался вокруг проблем моделирования в биологии, физиологии, медицине и этологии. Достижения этой школы в области индуктивного формирования понятий, моделей зрения, моделей роста тканей и коллективного поведения во многом не превзойдены западной наукой и по сей день.

Целый ряд этапов становления и развития компьютерной техники прошел на глазах у ныне живущего поколения людей. Можно приводить огромное количество фактов из истории вычислительной техники и мы не погрешим против истины, если скажем, что это тема для написания не одной большой книги.

В качестве краткой систематизации тех фактов, которые происходили в данной сфере исследователи предложили классифицировать начальный этап истории развития вычислительной техники в виде поколений ЭВМ. Всего таких поколений было выделено пять. Основным признаком для отнесения компьютеров к тому или иному поколению были электронные элементы, на базе которых они изготавливались.

К первому поколению принято относить компьютеры, созданные на базе радиоламп. Эти ЭВМ занимали целые машинные залы, потребляли огромное количество электроэнергии, были достаточно медленными и ненадежными. Годы к которым относят развитие первого поколения - 40-50е годы XX века.

Ко второму поколению относят компьютеры, разработанные на базе транзисторов. Эти ЭВМ могли размещаться уже в одном помещении, не требовали водяного охлаждения и были более надежными в работе. Второе поколение принято ассоциировать с началом 60-х годов XX века

Третье поколение обязано своим возникновением появлению интегральных схем (микросхем). Их еще называют интегральными схемами средней и большой степени интеграции - СИС и БИС. Характеристиками данного поколения, как вы понимаете, являлись уменьшение размеров и увеличение производительности и надежности ЭВМ. По времени - это середина 60-х начало 70 х годов XX века.

Четвертое поколение - это СБИС (сверхбольшие интегральные схемы) или их еще стали называть микропроцессорами (Начало и середина 70-х). Поистине появление микропроцессоров стало переломным моментом в истории развития компьютерной техники. Относительно недорогие и низко производительные микропроцессоры стали основой для разработки персональных компьютеров. Т.е. с этого момента вычислительная техника перестала быть уделом только профессионалов и шагнула буквально «в каждый дом» (что мы сейчас и наблюдаем).

Пятое поколение ЭВМ стоит особняком, так как его связывают с принятой в Японии государственной программой по разработке интеллектуальных компьютеров (компьютеров с «человеческим» интерфейсом). Это была амбициозная программа, предпосылкой возникновения которой, стали успехи Японии на мировом рынке бытовой электроники. Это породило желание догнать и перегнать американцев и в области компьютерной техники. К реализации упомянутой выше программы были привлечены большое количество молодых и перспективных инженеров и программистов, большое количество финансовых средств, но, по большому счету, те продукты, которые были разработаны, в результате не выдержали конкуренции с разработками частных фирм. Очевидным плюсом программы, с которым соглашаются все эксперты, стало воспитание целого поколения грамотных специалистов в области электроники и программирования, которые в последствии стали движущей силой Японской экономики.

Современная тенденция в развитии компьютерной техники, на наш взгляд, лежит в интеграции вычислительных и, самое главное, управляющих устройств в жизнь любого человека. Это и кухонные комбайны, интеллектуальные холодильники, стиральные машинки, технологии «умного» дома, автоматические диммеры и счетчики электроэнергии, контроллеры газоперекачивающих станций и газовых котлов, электронная начинка многочисленных теплопунктов. Контроллеры, управляющие солнечными батареями. Автомобильная электроника. Климат контроль в жилых и офисных помещениях. Многочисленные медийные и телекоммуникационные устройства. Молодое поколение, пожалуй, уже не мыслит себя без различных нетбуков, смартфонов и других гаджетов. Компьютеры, в том или ином виде, сейчас можно встретить практически во всех учреждениях: в школах, детсадах, университетах, офисах, на строительных площадках, в силовых структурах, в медицине, на транспорте. Электроника в медицине позволяет уже сейчас использовать искусственные органы, начиная от протезов конечностей и заканчивая внутренними органами. Особым образом в этом ряду можно отметить военную технику – высокоточное оружие, разрабатываемые робототехнические боевые и вспомогательные системы.

Электроника и компьютерная техника все большую роль стала играть в жизни не только отдельного человека, но и общества в целом. Это системы

локальной и глобальной безопасности. Космические системы мониторинга погоды, предсказания стихийных бедствий и землетрясений, Это и системы глобальной и локальной навигации. Это и «паутина» всемирной сети Интернет, которая стала не только кладезем информации и большой поисковой системой, но и просто культурным явлением, посредством которого люди могут общаться, показывать свои таланты, обмениваться мнениями. Это и моментальные платежные системы. Это, можно сказать, и коллективный разум, позволяющий объединять разработчиков, работающих над одним проектом и находящихся в разных точка Земного шара.

Все перечисленные выше достижения стали возможными благодаря уменьшению, удешевлению и повышению производительности компьютеров. Анализируя процесс развития компьютерной техники, были замечены интересные закономерности.

В 1965 году (через шесть лет после изобретения интегральной схемы) один из основателей Intel Гордон Мур обнаружил интересную закономерность. Появление новых типов микросхем наблюдается спустя примерно год после создания их предшественников. При этом, количество транзисторов, содержащихся в каждой новой микросхеме возрастает приблизительно вдвое. Г. Мур пришел к выводу о том, что при сохранении этой тенденции мощность вычислительных устройств за относительно короткий промежуток времени может вырасти экспоненциально. Это наблюдение получило название — закон Мура.

В 1975 году Гордон Мур внёс в свой закон коррективы, согласно которым удвоение числа транзисторов будет происходить каждые два года.

По поводу эффектов, обусловленных законом Мура, в восьмом номере за 1983 год журнала «В мире науки» (русское издание «Scientific American») было приведено такое сравнение:

«Если бы авиапромышленность в последние 25 лет развивалась столь же стремительно, как промышленность средств вычислительной техники, то сейчас самолёт Boeing 767 стоил бы 500 долл. и совершал облёт земного шара за 20 минут, затрачивая при этом пять галлонов (~18,9 л) топлива. Приведенные цифры весьма точно отражают снижение стоимости, рост быстродействия и повышение экономичности ЭВМ».

В 2007 году Мур заявил, что закон, очевидно, скоро перестанет действовать из-за атомарной природы вещества и ограничения скорости света.

15. Элементная база компьютерной техники

Чтобы лучше понять принципы функционирования компьютерных систем, остановимся на базовых элементах, которые лежат в основе всех разработок вычислительной техники.

Выше, при рассмотрении элементарных булевых функций, мы уже рассматривали логические вентили, теперь рассмотрим устройства с памятью и некоторые полезные суперпозиции логических вентилей: двоичные сумматоры, мультиплексоры, демультиплексоры, шифраторы, дешифраторы и т.п.

Триггер – простейшее последовательное устройство, которое может длительно находиться в одном из нескольких возможных устойчивых состояний и переходить из одного в другое под воздействием входных сигналов. Триггер – электронное устройство, реализующие хранение одного бита информации. Любой из триггеров может быть реализован набором элементов И-НЕ или ИЛИ-НЕ, путем введения обратных связей.

RS-триггер— триггер, который, если реализован на элементах И-НЕ (Рис.15.1), сохраняет своё предыдущее состояние при единичных входах и меняет своё выходное состояние при подаче на один из его входов нуля. Если RS-триггер реализован на элементах ИЛИ-НЕ, то хранение происходит при нулевых состояниях, а изменение – при подаче 1.

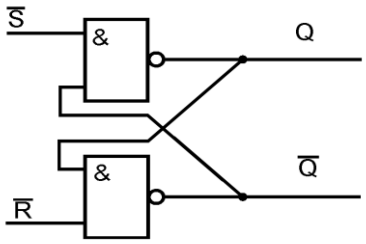


Рис.15.1

D-триггер – триггер, который запоминает состояние входа при подаче синхроимпульса и выдаёт его на выход. D-триггеры имеют, как минимум, два входа: информационный и синхронизации. Ниже (Рис.15.2) показана схема D-триггера, собранного на элементах И-НЕ.

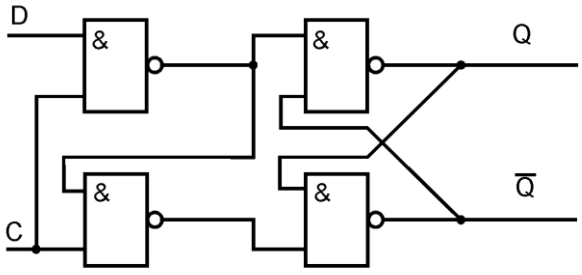


Рис.15.2

Различают синхронные и асинхронные триггеры. Промышленностью выпускаются микросхемы, содержащие от 2 до четырех триггеров с возможностью синхронной и асинхронной установки. Выделяют следующие типы триггеров: RS- триггеры, D-триггеры, JK-триггеры, T-триггеры.

Регистр – это набор триггеров с возможностью синхронной параллельной записи и считывания, асинхронной начальной установки, в некоторых случаях, последовательного ввода-вывода.

Счетчики – это устройства, используемые, в основном, для деления частоты входного сигнала. В процессе деления частоты на выходных полюсах счетчиков отображаются в параллельном коде возрастающая или убывающая двоичные или двоично-десятичные последовательности. Максимальный коэффициент деления зависит от разрядности счетчика. Счетчики могут иметь возможность параллельной установки и асинхронного сброса. Обычно счетчики реализуются на базе JK и T-триггеров.

Мультиплексоры – устройства, которые используются для коммутации нескольких информационных каналов в один. Реализуются на обычной логике.

Шифраторы – устройства, использующиеся для кодирования номера активизированного входного полюса микросхемы для уменьшения количества линий необходимых для передачи информации.

Дешифраторы – устройства, активизирующие выходной полюс или полюсы, которые соответствует коду, подаваемому на вход устройства. Дешифраторы также используются для управления сегментными индикаторами.

16. Вычислительные устройства

В простейшем случае модель компьютера может быть представлена в виде процессора, памяти, и соединяющей их шины (Рис.16.1).

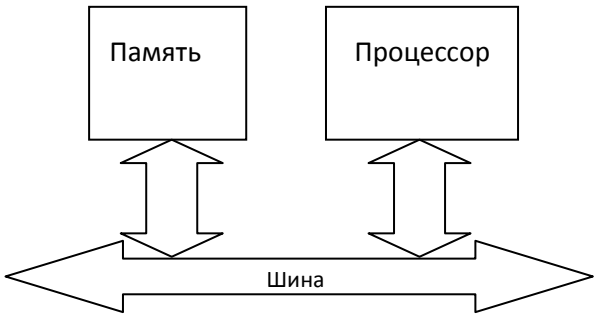


Рис 16.1

Память хранит команды и данные. Процессор выполняет команды и обрабатывает данные. Алгоритм функционирования в этом случае выглядит следующим образом. После включения питания специальное устройство формирует начальный адрес, с которого будет прочитана первая команда. Процессор читает из памяти код, который он воспринимает как команду, затем он выполняет команду и готовится к выполнению следующей команды. Можно сказать, что процессор может выполнять только два действия: выбирать команды из памяти и выполнять их. Говоря более строго, возможно и третье действие – стандартная реакция на прерывания. Немного усложнив

первоначальную модель, вводят внешние устройства (устройства ввода – вывода), контроллеры, т.е. устройства управления внешними устройствами и устройства обработки прерываний процессора.

16.1. Модель Фон-Неймана

В полученной модели все структурные блоки соединяются через единую системную магистраль. Если данная модель, вдобавок ко всему, будет удовлетворять четырем приведенным ниже условиям, то мы имеем классическую модель компьютера предложенную Дж.Фон-Нейманом и носящую название Принстонской (Рис.16.2).

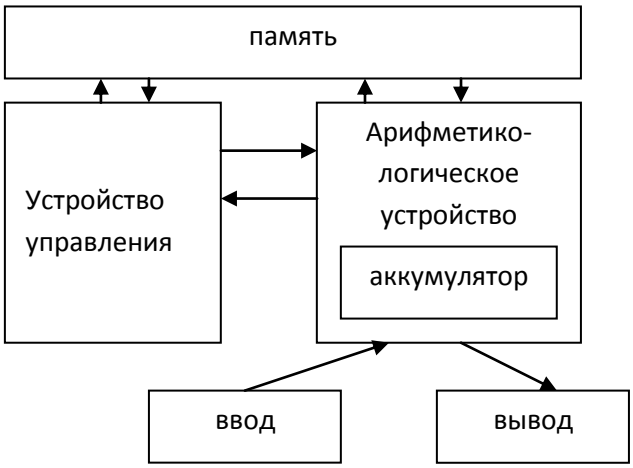


Рис.16.2

Условия:

- Наличие единого вычислительного устройства, включая процессор, средства передачи информации и память;
- Линейная структура адресации памяти, состоящей из слов фиксированной длины;
- Низкий уровень машинного языка, команды которого осуществляют простые операции над простыми операндами;
- Централизованное последовательное управление.

В отличии от Принстонской архитектуры микроконтроллеры, с которыми приходится иметь дело, имеют архитектуру Гарвардскую (Рис.16.3), т.е. отдельную память данных и программ.

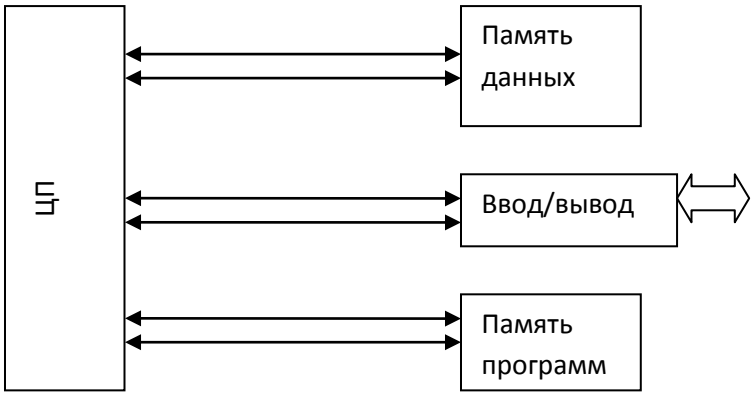


Рис.16.3

Этот факт и факт использование *RISC (redused instruction set command)* – сокращенного набора команд позволяет существенно увеличить быстродействие микроконтроллера, не повышая тактовую частоту кристалла. Отметим, что противоположностью RISC архитектуре считаются процессоры, имеющие архитектуру *CISC (complete instruction set command).*

16.2. Аппаратные средства компьютера

Рассмотрим подробнее, как функционирует простой среднестатистический компьютер. Управляющая работой ЭВМ программа перед началом выполнения загружается в ОЗУ (оперативное запоминающее устройство или, другими словами, оперативную память (ОП)). Адрес первой выполняемой команды передается микропроцессору и запоминается в счетчике команд. Начало работы процессора заключается в том, что адрес из счетчика команд (в котором всегда хранится адрес очередной команды) выставляется на шину адреса системной магистрали. Одновременно на шину управления подается команда – выборка из оперативной памяти. Получив с шины управления команду, контроллер оперативной памяти считывает адрес с шины адреса, находит ячейку с этим адресом и выставляет ее значение на шину данных. Процессор, получив сигнал об окончании работы контроллера оперативной памяти, вводит число с шины данных на свою внутреннюю шину и через нее посылает введенную информацию в регистр команд. В регистре команд полученная команда разделяется на кодовую и адресную части (посредством дешифрации команд). Код команды поступает в блок управления для выработки сигналов, настраивающих МП (Микропроцессор, процессор) на выполнение заданной операции и определение адреса следующей команды (который сразу же поступает в счетчик команд). Адресная часть команды поступает на шину адреса системной магистрали, а на управляющей шине выставляется сигнал на чтение из ОП. Выбранная из ОП информация через шину данных поступает на внутреннюю магистраль МП, с которой вводится в АЛУ (арифметико–

логическое устройство). На этом заканчивается подготовка МП к выполнению операции и начинается процесс ее выполнения в АЛУ.

АЛУ выполняет арифметические операции и операции, связанные с математической логикой и принятием решений (команды переходов по условию). Большинство АЛУ могут производить сложение, вычитание, сдвиги вправо и влево. Умножение и деление программируется в блоке управления.

Еще несколько обобщающих слов о процессоре. И так, процессор – это совокупность устройства управления (УУ), арифметико-логического устройства (АЛУ) и связанных с ним регистров. Процессор организует процесс обработки информации путем выборки и последовательного выполнения команд программы, находящейся в памяти компьютера.

АЛУ состоит из арифметико-логической цепи (главным образом двоичного сумматора, который может производить сложение, вычитание и сдвиг данных). Для выполнения операции данные помещаются в аккумулятор и регистр данных. Результат операции помещается в регистр данных.

Устройство управления содержит *Регистр* команд и Д*ешифратор* команд, а так же *Блоки синхронизации и управления.*

Регистры:

Аккумулятор используется для промежуточного хранения и накопления данных во время выполнения операции

Регистр состояния хранит флаги результата операций: переполнение, сравнения с нулем и т.п.

Указатель стека содержит адрес ячейки памяти данных, которые хранятся в стеке.

Теперь несколько слов скажем о прерываниях. *Прерывания* – это готовые процедуры (подпрограммы), которые компьютер вызывает для выполнения определенной задачи. Существуют аппаратные и программные прерывания. Аппаратные прерывания инициируются аппаратурой, либо с системной платы, либо с карты расширения. Они могут быть вызваны сигналом микросхемы таймера, сигналом от принтера, нажатием клавиши на клавиатуре и множеством других причин. Аппаратные прерывания не координируются работой программного обеспечения. Когда вызывается прерывание, то процессор оставляет свою работу, выполняет прерывание, а затем возвращается на прежнее место. Для того, чтобы иметь возможность вернуться в нужное место в программу, адрес этого места запоминается в стеке, вместе с регистром флагов. Затем в счетчик команд загружается адрес программы обработки прерывания и ей передается управление. Программы обработки прерываний всегда завершаются инструкцией возврата из прерывания, которая завершает процесс, начатый прерыванием, возвращая старые значения адреса основной программы и регистра флагов, тем самым давая программе возможность

продолжить выполнение из того же состояния. При реализации прерываний в младших семействах микроконтроллеров сохранение контекста и адреса возврата из прерывания ложится на плечи программиста.

С другой стороны, программные прерывания фактически ничего не прерывают. На самом деле это обычные процедуры, которые вызываются вашими программами для выполнения рутинной работы, такой как прием нажатия клавиши на клавиатуре или вывод на экран. Однако эти подпрограммы содержатся не внутри вашей программы, а в операционной системе и механизм прерываний дает вам возможность обратиться к ним. Программные прерывания могут вызываться друг из друга. Например, все прерывания обработки ввода с клавиатуры DOS используют прерывания обработки ввода с клавиатуры BIOS для получения символа из буфера клавиатуры.

Адреса программ прерываний называют векторами. Каждый вектор имеет длину четыре байта. Младшие 1024 байт памяти IBM PC совместимом компьютере содержат вектора прерываний, таким образом имеется место для 256 векторов. Вместе взятые они называются таблицей векторов.

Выше мы рассмотрели элементарные принципы, лежащие в основе функционирования аппаратных средств вычислительной техники. Однако, очевидно, что без программного обеспечения весь спектр вычислительных систем, какой бы замечательной архитектурой они не обладали – это только набор безжизненного «железа» (hardware).

17. Программное обеспечение компьютера

Жизнь в компьютеры могут вдохнуть только программы. Те люди, кто так или иначе сталкивался с программным обеспечением, имеют представление о том, какое существует разнообразие программ. Соответственно, существует и достаточно большое количество подходов к классификации программного обеспечения. Например, по способу реализации или вызова программ можно было бы разделить их на микропрограммы, хранящиеся непосредственно в процессоре и реализуемые, в основном, аппаратно; аппаратные и программные прерывания; и, собственно, программы, написанные на языках различного уровня.

На заре компьютерной эры первые компьютеры программировались простой коммутацией панелей и проводов. Для записи значений в регистры применялись обыкновенные тумблеры. Далее стали писать программы, используя машинные коды. Прорывом в программировании стало появление внешних носителей (перфокарт, первыми применили в фирме IBM), способных хранить программы вне оперативной памяти компьютера.

Разработка трансляторов – специальных программ, переводящих команды с языка программиста (языка удобного для человека) на язык машинных кодов, сыграла существенную роль в развитии компьютерных программных средств. Первым из таких "удобных" языков программирования стал язык Ассемблер.

Точнее было бы сказать Ассемблеры, так как каждый тип процессора требовал «своего» Ассемблера.

Далее появились линковщики и компиляторы, которые давали возможность прилинковывать (пристыковывать, присоединять) к разрабатываемой программе уже готовые модули библиотек (наборов часто используемых подпрограмм).

Если транслятор сначала переводил готовую программу в машинные коды, а затем ее исполнял, то интерпретаторы позволяли производить «построчное» преобразование и исполнение программы. Исполняемые программы, которые получались посредством применения транслятора или интерпретируемые программы привязаны к конкретному типу процессора. Решить задачу совместимости разных аппаратных платформ позволило изобретение так называемого байт-кода (промежуточного кода – универсального для различных платформ), который преобразовывался в интерпретируемый или исполняемый код для конкретной платформы уже на конечном этапе подготовки программы к выполнению. В конце девяностых годов XX века появился еще один вариант исполняемой программы – это выполнение программы в рамках виртуальной машины, которая полностью эмулировала операционную среду реального компьютера.

Сейчас каждый компьютер или смартфон оснащен операционной системой. Однако, что такое операционная система и зачем она нужна знает далеко не каждый пользователь. В начале компьютерной эры, когда на одном процессоре выполнялась одна программа особых проблем с вычислительными ресурсами не было. Когда же возникла острая необходимость разделять вычислительные ресурсы, оптимизировать «машинное время», стали думать как более эффективно использовать время работы процессора.

И такое решение было найдено. Если изначально исполняемую программу загружали в память в виде машинных кодов вручную, то с появлением трансляторов появилась необходимость загружать в память сначала транслятор, затем программу, после этого давать команду на трансляцию, и на выполнение программы. Если программы были написаны на разных языках программирования, то приходилось загружать сначала один транслятор, затем другой, затем опять первый. Достаточно большим шагом вперед оказалась реализация идеи использования пакетного способа загрузки и обработки программ. Т.е. формировался пакет программ, написанных на одном языке программирования, загружался транслятор для этого языка, затем загружались программы, транслировались и по очереди отправлялись на исполнение. Для того чтобы управлять процессом выполнения программы, перед загрузкой программы формировались так называемые рабочие карты (Job карты), в которых описывались приоритеты исполняемых программ и т.п.

Все шаги, которые были описаны выше – это только верхняя часть «айсберга». На самом деле программы еще должны были взаимодействовать с

внешними устройствами – выводить результаты расчетов на печатающее устройство, отображать результаты промежуточных действий на дисплее и т.п. Для автоматизации этих и других «рутинных» операций по взаимодействию пользователя с «компьютерным железом» и были разработаны операционные системы.

Ядром, утилитами, сервисными программами большинства операционных систем выполняются следующие задачи:

- Начальный загрузка программ (начальный загрузчик);
- Управление аппаратными средствами компьютера (драйверы);
- Управление файлами, хранение информации на жестком диске разграничение доступа к файлам и каталогам (файловая система);
- Распределение оперативной памяти, защита памяти (диспетчер программ (диспетчер памяти));
- Задачи аутентификации.

Приблизительная последовательность действий при выполнении различных задач под управлением операционной системы состоит из следующих шагов:

- Загрузка операционной системы;
- Управление устройствами - драйверы;
- Работа с файлами;
- Запуск пользовательских программ (прикладных программ);

Ниже приведена одна из возможных классификаций операционных систем. Согласно этой классификации операционные системы можно разделить на:

- Однозадачные;
- Однозадачные с фоновой печатью;
- Многозадачные;
- Локальные;
- Сетевые;
- Реального времени;
- Мобильные;

Из доброй сотни известных операционных систем можно выделить всего несколько таких, с которыми сталкиваются непрофессиональные пользователи в повседневной жизни:

- Семейство Microsoft (Windows, Windows CE, Windows Mobile)
- Семейство Apple (Mac OS, Apple iOS)
- Семейство UNIX (Linux, Google Android)

Во время взаимодействия с компьютером даже простой пользователь так или иначе сталкивается с огромным количеством программ. К одним он обращается сам, а о существовании других он даже не подозревает. В таблице

17.1 приведена одна из возможных классификаций существующих компьютерных программ:

Таблица 17.1

Типы программ	*Назначение и примеры программ*
Оболочки облегчают работу с файлами, каталогами, сетевыми и другими ресурсами	*Norton Commander;* *Volkov Commander;* *Midnight Commander;* *Windows Commander;* *FAR;* *Проводник*
Сервисное ПО; *Утилиты*	*Управление питанием;* *Энергосистема;* *Хранители экрана;* *Работа с диском;* *Дефрагментация;* *Перераспределение дискового пространства;* *Отдача ресурсов в пользование;* *Управление доступом;* *Заведение пользователей;* *Диагностика дисков;* *Диагностика процессора;* *Проверка быстродействия;* *Антивирусное ПО;* *Архиваторы;*
Сетевое обеспечение компьютера	*Сетевые утилиты для настройки сети;* *Браузеры*
Мультимедийное ПО	*Просмотровщики фильмов (кодеки);* *Проигрыватели музыки;* *Музыкальные и видео редакторы;* *Управление аппаратурой*
Офисные пакеты	*Текстовый процессор;* *Электронные таблицы;* *Подготовка презентации;* *Органайзеры (подготовка расписаний);* *«Напоминалки»*
Издательские системы	*Редактирование материала;* *Верстка*

Графические пакеты	*Векторные;* *Растровые;* *Трехмерная графика*
СУБД	*Системы управления базами данных* *Базы данных (хранилища данных);* *Графические оболочки и интерфейс взаимодействия;* *Драйверы и библиотеки доступа;* *Система безопасности*
CAD-системы *Computer-Aided Design. Системы автоматического проектирования САПР*	*Разработка и моделирование печатных плат и электронных схем;* *Разработка и моделирование домов;* *Разработка и моделирование конструктивных элементов*
Поисковые и справочные системы	*Телефонные справочные системы;* *Поиск в Интернете*
Коммуникация	*IP-телефония.* *Скайп;* *Потоковое телевидение;* *Потоковое радио;* *Системы электронной почты* *Почтовые серверы* *Почтовые клиенты* *Локальные клиенты* *WEB клиенты*
Системы электронной коммерции	*Internet магазины* *Системы продажи билетов* *Различные услуги* *Электронные деньги* *Карточки* *Кошельки* *Платежные системы*
Системы разработки программ	*Редакторы* *Компиляторы* *Отладчики*
Геофизические системы	*Карты и фотографии Земли* *Карты Луны и Солнца* *Системы мониторинга и прогноза*

	погоды
Системы защиты ПК.	*Антивирусное ПО* *Системы шифрования и электронной подписи* *FierWall, Брандмауэр* *Proxy*
Системы дистанционного и локального обучения	
Математические пакеты	
Электронные игры и Симуляторы	

17.1. Краткая история языков программирования

Если быть очень кратким, то историю развития языков программирования можно попробовать изобразить в виде шести пунктов:

1. Механические устройства и станок Шиккарда – программирование при помощи перфоленты;

2. Табулятор Г. Холлерита – использование перфокарт для обработки результатов переписи населения;

3. Эниак – программирование путем изменения конфигурации подключения различных блоков и коммутация переключателей;

4. Серийные компьютеры, начиная с первого поколения – программирование в машинных кодах (пример выполнение команды):

```
1000  12737
1002  10
1004  256
```

5. Большие ЭВМ и персональные компьютеры – программирование на ассемблере (пример выполнение команды):

```
mov  #10,256
```

6. Практически все виды компьютеров – алгоритмические языки, где каждая строка кода могла бы автоматически транслироваться в цепочку машинных команд. Этим занимались три типа программ: трансляторы; компиляторы; исполнители (интерпретаторы)

К числу первых алгоритмических языков можно отнести *Фортран (FORTRAN – FORmula TRANslation*) транслятор формул. Производство фирмы

IBM 1954 г. В нашу страну пришел с IBM 360, IBM 370 (ЕС ЭВМ - клон). Вариации Fortran-II, Fortran IV, Fortran 77, Fortran 90

Алгол–60 – противовес Фортрану – как объединение усилий европейских ученых (академическая и вузовская наука)

Бейсик многое взял от Фортрана; Паскаль от Алгола

Первое поколение – день рождения Бейсика 1 мая 1964г. Дартмутский колледж (США). Интерпретатор создан студенческим коллективом во главе с профессором Джоном Кемени (J.G.Kemeny) и Томасом Куртцем (T.E.Kurtz). Название: Beginner's All -pupose Symbolic Instruction Code – «Многоцелевой код (язык) символических инструкций для начинающих». Basic открыл эру диалогового программирования.

Становление в нашей стране Бейсика – первого в мире пошагового компилятора для ЭВМ М-20, созданной в Горьковском университете в 1969 году.

Второе поколение Бейсик систем – BASIC-80 разработан в середине 70-х для процессоров Intel 8080 и Z-80. С этого интерпретатора началась карьера Билла Гейтса.

Третье поколение: Turbo Basic фирмы Borland и Quick Basic фирмы Microsoft.

Четвертое поколение: Visual Basic. 1993 год - VBA (Visual Basic for Applications) для MS Office, VBS Visual Basic Script.

Pascal был разработан в 1968 году. Автор Никлаус Вирт - профессор института при Швейцарской высшей технической школе. В 1970 году был выпущен первый компилятор, названный в честь известного французского математика Блеза Паскаля, который в 19 лет собрал первую суммирующую машину. Язык отличается жесткими требованиями к описанию переменных и констант, имеет блочные конструкции. В 1982 году выпущен стандарт ISO, однако язык использовался в основном в университетах. Н.Вирт отказался от работ над стандартом и переключился на проект Модула.

В 1984 г. молодой француз Филипп Кан привез в США необычный по скорости компилятор Turbo Pascal для IBM совместимых ПК и начал торговать им по цене $49,95. В то время интерпретатор Basic стоил 500$. Последняя известная версия TP 7.0 и Borland Pascal для Win. На базе Object Pascal был разработан язык *Delphi.*

Язык Си был разработан в 1972 году сотрудником Bell Laboratories отделения телефонной компании AT&T Денисом Ритчи одним из первых пользователей UNIX. Язык Си задумывался ни как алгоритмический язык, а как инструмент развития OS Unix для создания новых утилит.

Электронные таблицы MS Excel разрабатывались для упрощения обработки больших объемов числовых данных, для выполнения рутинных операций и наглядного представления результатов вычислений. Неплохим продуктом из этой области был *Super Calc*. Даже неподготовленный пользователь мог, пользуясь простыми формулами, выполнять бухгалтерские, статистические операции. VBA позволяет максимально использовать готовые интерфейсные возможности Windows с максимально доступным интерпретатором. Идеология VBA заключается в том, что основными при реализации процедур автоматизации является не программа, а документ в котором хранится программа, хотя программы могут храниться и отдельно от документа. VBA позволяет самообучаться при включении автоматической записи макроса. Мы будем недалеки от истины, если скажем, что данная платформа и язык возглавляют мировой рейтинг среди средств программной автоматизации.

Язык C++ возник в начале 1980-х годов, когда сотрудник фирмы *Bell Labs Бьёрн Страуструп* придумал ряд усовершенствований к языку *Си* под собственные нужды. Часть идей и конструкций для языка C++ были взяты из языков Симула и BCPL. Каждый из этих языков имел недостатки, которые не позволяли использовать его полноценно для определенных нужд. Так, язык Симула имел полезные для разработки большого программного обеспечения возможности, но работал слишком медленно. Язык BCPL был достаточно быстрым, но по сути был слишком близок к языкам низкого уровня и не подходил для разработки большого ПО. Страуструп решил дополнить язык *Си* (преемник BCPL) возможностями, имеющимися в языке Симула. Страуструп добавил к языку Си возможность работы с классами и объектами. В результате практические задачи моделирования оказались доступными для решения как с точки зрения времени разработки (благодаря использованию Симула-подобных классов), так и с точки зрения времени вычислений (благодаря быстродействию Си). В первую очередь в Си были добавлены классы (с инкапсуляцией), наследование классов, строгая проверка типов, inline-функции и аргументы по умолчанию. Ранние версии языка, первоначально именовавшегося «C with classes» («Си с классами»), стали доступны с 1980 года.

Delphi (Дельфи) — среда программирования, использующая язык Object Pascal, разработанный фирмой Borland и изначально реализованный в её пакете Borland Delphi, от которого и получил в 2003 году своё нынешнее название. По сути является наследником языка Pascal с объектно-ориентированными расширениями. Свою историю язык Object Pascal ведет с первых Turbo Pascal`ей разработанных Никлаусом Виртом.

Когда компанией Microsoft был совершен переворот в области прикладного программирования – впервые создана визуальная система разработки прикладных программ, реализованная в Visual Basic, к новому

стандарту стали приобщать все языки программирования. Не избежал своей участи и Pascal. Компанией Borland было пересмотрено и переработано множество функций, создано много оригинальных конструкций. В итоге изменения языка Pascal оказались столь колоссальными, что разработчики создали, можно сказать, новый язык программирования Delphi.

Delphi оказал огромное влияние на создание концепции языка C# для платформы .NET. Многие его элементы и концептуальные решения вошли в состав C#. Одной из причин называют переход Андерса Хейлсберга, одного из ведущих разработчиков Дельфи, из компании Borland Ltd. в Microsoft Corp.

Говоря о языках программирования нельзя не упомянуть о линейке функционально-декларативных языков программирования, начатых от Lisp. Сейчас эти языки менее популярны, нежели традиционные императивные, однако идеи, заложенные в них, оказали существенное влияние и на узкоспециализированные направления, (такие как язык PostScript), и на телекоммуникационные направления (язык Erlnag) и на высоко академические исследования (язык Haskell). Некоторые исследователи считают, что за ними будущее программирования.

18. Системы управления базами данных

Языки программирования предназначены для реализации различных алгоритмов, реализуемых и выполняемых на компьютерах. Однако конструкции классических языков программирования становятся мало пригодными, когда речь заходит о хранении, обработке и поиске больших объемов информации. Все эти вопросы решаются с помощью систем управления базами данных (СУБД).

Для лучшего понимания того как функционируют СУБД, как можно с ними взаимодействовать, необходимо усвоить некоторые понятия.

Базой данных (БД) называют поименованную сущность, поименованную совокупность данных, относящаяся к определенной предметной области.

СУБД - *комплекс программных и языковых* средств, необходимых для *создания*, *обработки и поддержании* баз данных в актуальном состоянии. СУБД имеют средства для *эффективного хранения* информации, средства *защиты* данных от случайной потери/порчи, средства *экономного использования* ресурсов, средства *быстрого поиска* информации.

Примеры некоторых СУБД: Access, SQL Server, Oracle, IBM DB/2, Interbase, Informix, MySQL, SQLite.

Некоторая терминология баз данных:

Предметная область – часть реального мира, информация о которой моделируется средствами баз данных.

Модель данных – это описание предметной области. Данная модель включает в себя определения сущностей и атрибутов сущностей. *Сущностями* обычно называют *классы* или совокупности объектов, имеющих одинаковое назначение, обладающих одинаковыми свойствами. Например, класс – Студенты; класс – Преподаватели; класс – Родители; класс – Автомобили; класс – Бытовая техника. У каждой сущности есть какие-то характерные *признаки*, по которым один экземпляр сущности можно отличить от другого экземпляра сущности, такие *признаки* или свойства называют *атрибутами*. Например, автомобили отличаются друг от друга по марке автомобиля, объему двигателя, заводу производителю, году выпуска, цвету и.т.п. Студенты могут характеризоваться (отличаться друг от друга), фамилией, именем, отчеством, годом рождения, специальностью, номером группы в которой обучаются, городом из которого приехали и т.п. Характерные или отличительные признаки могут выражаться простыми словами (цвет – красный), а могут являть собой и целые сущности (фирма производитель – *Дженерал моторс, Тойота, Фольксваген)*, которые имеют свои наборы характеризующих их атрибутов.

Многие современные БД являются реляционными базами данных.

18.1. Принципы реляционной модели

Реляционная модель данных была разработана сотрудником фирмы *IBM Эдгаром Франком Коддом (Edgar Frank Codd)* и опубликована в 1970 г в работе «A Relational Model of Data for Large Shared Data Banks». Реляционная модель определяет способ представления данных (структуру данных), методы обеспечения целостности данных, и операции, которые можно выполнять с данными.

Основные принципы реляционных баз данных:

– каждой сущности в реляционной модели данных соответствует отношение (*relation*) - упорядоченная организация данных, определенная в виде строк и столбцов. Чаще вместо слова «отношение» - используют просто слово таблица («набор записей», или набор результатов - result set). Именно от этого и происходит термин «реляционные базы данных»;

– каждому экземпляру сущности в отношении (таблице) соответствует кортеж или запись (строка таблицы);

– мощность отношения – число кортежей в отношении (проще говоря, число записей или строк в таблице);

– каждому атрибуту сущности соответствует столбец или поле в отношении (таблице);

– размерность – это число атрибутов в отношении;

– каждое отношение можно разделить на две части – заголовок и тело. На простом языке заголовок отношения – это список (названия) столбцов (полей), а тело – это сами записи (кортежи);

– каждый атрибут характеризуется таким понятием как «домен» или набор данных (тип данных). Понятие Домен можно интерпретировать как множество *из которого «черпают» свои значения атрибуты.*

Как мы уже отмечали выше, сущности, информация о которых хранится в базе данных, могут характеризовать другие сущности. Т.е. сущность *автомобиль* может характеризоваться сущностью *завод производитель* или сущностью компания поставщик. Другими словами, одни сущности должны быть связаны с другими сущностями. Поэтому появляется необходимость каким-то образом связывать таблицы, из которых состоит база данных. Такой способ виртуального связывания таблиц существует и обеспечивается наличием общих для связываемых таблиц ключевых полей. С одной стороны (для одной таблицы) это основные или первичные ключевые поля (*primary key),* а с другой стороны - внешние ключевые поля или *(foreign key*). Отметим, что *основной ключ (primary key)* обязательно должен быть уникальным, т.е. в этом поле не должно быть повторяющихся значений. Также следует отметить, что в качестве основного ключа может выступать некоторая совокупность простых полей.

Структуры связей между таблицами принято делить на *связи 1:1, 1:M, M:1, M:M.*

Связь между таблицами называют связью 1:1 в том случае, если одной записи в одной таблице соответствует 1 запись в другой таблице. В качестве примера можно привести связь таблицы «студенты» и таблицы «зачетная_книжка». Одному студенту однозначно соответствует одна зачетная книжка. Обычно такие типы связи на практике реализуются путем добавления к искомой таблице дополнительного поля.

Часто используемыми типами связей между таблицами являются *1:M* один ко многим или *M:1* - многие к одному. Такой тип связи используется в том случае, когда одной записи в одной таблице соответствует много записей в другой таблице. Например, у одного студента может быть несколько телефонных номеров (телефоны разных операторов связи) . В этом случае к таблице «Телефон» добавляется ключевое поле «Key_stud», которое является внешним ключом для данной таблицы и первичным ключом для таблицы Stud. В него заносятся значения поля «Key_stud» (первичного ключа таблицы Stud), соответствующие экземпляру сущности «студент», имеющему соответствующие телефонные номера.

Еще один распространенный тип связей – связь многие ко многим *M:M.* Т.е одной записи в таблице А может соответствовать несколько записей в таблице В и наоборот одной записи в таблице В может соответствовать несколько записей в таблице А. В качестве примера можно привести связи между сущностями «студент» и «родители». Каждому экземпляру сущности «студент» может соответствовать несколько экземпляров сущности «родитель». Другими словами, у студента может быть отец, мать, дедушки и бабушки. В то

же время у отца или матери могут быть несколько детей, которые учатся в данном учебном заведении. Для реализации связи M:M используют внешнюю таблицу, содержащую обычно три поля: основной ключ самой таблицы и внешние ключи связываемых таблиц.

18.2. Язык структурированных запросов SQL

Язык структурированных запросов *(Structured Query Language - SQL)* является промышленным стандартом языка баз данных, используемым ядром базы данных Microsoft. Ранним предшественником SQL был язык *SEQUEL*.

SQL обеспечивает команды как языка определения данных (data definition language - DDL) так и языка управления данными (data manipulation language - DML). Команды DDL позволяют создавать и определять новые базы данных, поля и индексы, в то время как команды DML позволяют формировать запросы сортировки, фильтрации и извлечения данных.

Ниже рассмотрим некоторые конструкции языка DML

SELECT – используется, чтобы извлечь записи из БД

*SELECT * FROM stud* – запрос на выбор всех данных из всех полей из таблицы stud

SELECT fio_stud FROM stud – запрос на выбор всех данных из поля fio_stud из таблицы stud

DELETE – удаляет из одной и более таблиц записи, удовлетворяющие условию WHERE

DELETE FROM stud WHERE key_stud = 22 – удаляет из таблицы stud запись, которой соответствует значение поля key_stud, равное 22

INSERT INTO - добавляет данные в таблицу

INSERT INTO town (name_town, region_town) VALUES ('Томск','Россия')

UDATE ... SET – обновляет все данные в таблице, удовлетворяющие условию WHERE

UPDATE stud SET fio_stud='Петров В.В.' WHERE fio_stud like 'Петров В.А.'

Данный запрос позволяет изменить у всех экземпляров сущности Студент, хранящихся в таблице stud, имеющих значение поля fio_stud, похожее на "Петров В.А." на значение "Петров В.В."

19. Экспертные системы

Если базы данных используются для хранения и поиска информации, то экспертные системы используют определенным образом структурированную информацию в качестве знаний и позволяют применять ее для принятия решений

Центральная парадигма интеллектуальных информационных технологий сегодня — это обработка знаний. Системы, ядром которых является база знаний или модель предметной области, описанная на языке сверхвысокого уровня, приближенном к естественному, называют интеллектуальными системами (ИС).

Наиболее распространенным видом ИС являются экспертные системы.

Экспертные системы (ЭС) — это наиболее распространенный класс ИС, ориентированный на тиражирование опыта высококвалифицированных специалистов в областях, где качество принятия решений традиционно зависит от уровня экспертизы, например, медицина, юриспруденция, геология, экономика, военное дело и др.

ЭС эффективны лишь в специфических «экспертных» областях, где важен эмпирический опыт специалистов.

Только в США ежегодный доход от продаж инструментальных средств разработки ЭС составлял в начале 90-х годов 300-400 млн. долларов, а от применения ЭС — 80-90 млн. долларов. Ежегодно крупные фирмы разрабатывают десятки ЭС типа «in-house» для внутреннего пользования. Эти системы интегрируют опыт специалистов компании по ключевым и стратегически важным технологиям. В начале 90-х гг. Появилась новая наука — «менеджмент знаний» *(knowledge management*), ориентированная на методы обработки и управления корпоративными знаниями

Современные экспертные системы — это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и распространяющие этот эмпирический опыт для консультирования менее квалифицированных пользователей. Разработка экспертных систем, как активно развивающаяся ветвь информатики, направлена на использование ЭВМ для обработки информации в тех областях науки и техники, где традиционные математические методы моделирования малопригодны. В этих областях важна смысловая и логическая обработка информации, важен опыт экспертов.

Приведем некоторые условия, которые могут свидетельствовать о необходимости разработки и внедрения экспертных систем

- нехватка специалистов, затрачивающих значительное время для оказания помощи другим;
- выполнение небольшой задачи требует многочисленного коллектива специалистов, поскольку ни один из них не обладает достаточным знанием;
- сниженная производительность, поскольку задача требует полного анализа сложного набора условий, а обычный специалист не в состоянии просмотреть (за отведенное время) все эти условия;
- большое расхождение между решениями самых хороших и самых плохих исполнителей;

• наличие конкурентов, имеющих преимущество в силу того, что они лучше справляются с поставленной задачей.

Подходящие задачи имеют следующие характеристики:

• являются узкоспециализированными;

• не зависят в значительной степени от общечеловеческих знаний или соображений здравого смысла;

• не являются для эксперта ни слишком легкими, ни слишком сложными. (Время, необходимое эксперту для решения проблемы, может составлять от трех часов до трех недель.)

Обобщенная структура экспертной системы представлена на Рис.19.1. Следует учесть, что реальные ЭС могут иметь более сложную структуру, однако блоки, изображенные на рисунке, непременно присутствуют в любой действительно экспертной системе, поскольку представляют собой стандарт de facto структуры современной ЭС.

В целом процесс функционирования ЭС можно представить следующим образом: пользователь, желающий получить необходимую информацию, через пользовательский интерфейс посылает запрос к ЭС; решатель, пользуясь базой знаний, генерирует и выдает пользователю подходящую рекомендацию, объясняя ход своих рассуждений при помощи подсистемы объяснений.

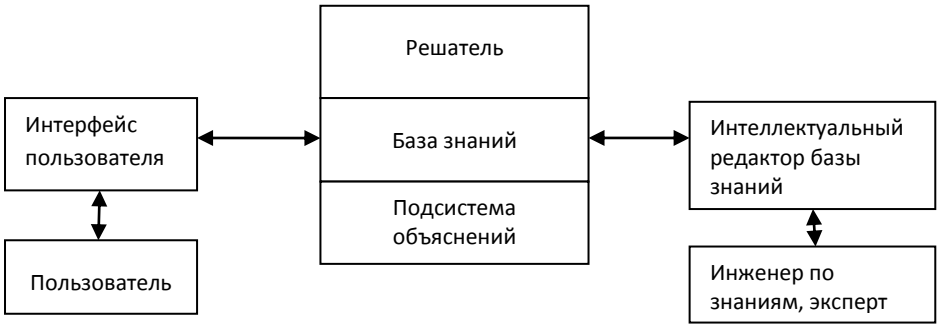


Рис.19.1

Пользователь — специалист предметной области, для которого предназначена система. Обычно его квалификация недостаточно высока, и поэтому он нуждается в помощи и поддержке своей деятельности со стороны ЭС.

Инженер по знаниям — специалист в области искусственного интеллекта, выступающий в роли промежуточного буфера между экспертом и базой знаний. Синонимы: когнитолог, инженер-интерпретатор, аналитик.

Интерфейс пользователя — комплекс программ, реализующих диалог пользователя с ЭС как на стадии ввода информации, так и при получении результатов.

База знаний (БЗ) — ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному). Параллельно такому «человеческому» представлению существует БЗ во внутреннем «машинном» представлении.

Решатель — программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в БЗ. Синонимы: дедуктивная машина, машина вывода, блок логического вывода.

Подсистема объяснений — программа, позволяющая пользователю получить ответы на вопросы: «Как была получена та или иная рекомендация?» и «Почему система приняла такое решение?» Ответ на вопрос «как» — это трассировка всего процесса получения решения с указанием использованных фрагментов БЗ, то есть всех шагов цепи умозаключений. Ответ на вопрос «почему» — ссылка на умозаключение) непосредственно предшествовавшее полученному решению) то есть отход на один шаг назад. Развитые подсистемы объяснений поддерживают и другие типы вопросов.

Интеллектуальный редактор БЗ — программа, представляющая инженеру по знаниям возможность создавать *БЗ* в диалоговом режиме. Включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок *(«help» — режим*) и других сервисных Средств, облегчающих работу с базой.

Основой экспертной системы является база знаний. В дополнение к приведенным выше определениям можно добавить следующую интерпретацию: Знания – это данные (факты) плюс правила вывода на данном множестве фактов

19.1. Представление знаний

При изучении интеллектуальных систем традиционно возникает вопрос — что же такое знания и чем они отличаются от обычных данных, обрабатываемых на ЭВМ. Можно предложить несколько рабочих определений, в рамках которых это становится очевидным.

Данные — это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

Знания — это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и

профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.

19.1.1. Семантические сети

Термин семантическая означает «смысловая», а сама семантика — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, то есть наука, определяющая смысл знаков.

Семантическая сеть — это ориентированный граф, вершины которого — понятия, а дуги — отношения между ними.

В качестве понятий обычно выступают абстрактные или конкретные объекты, а отношения — это связи типа: «это» (*«AKO — A-Kind-Of», «is»),* «имеет частью» *(«has part»),* «принадлежит», «любит». Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

- класс — элемент класса (цветок — роза);
- свойство — значение (цвет — желтый);
- пример элемента класса (роза — чайная).

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, отражающей поставленный запрос к базе.

Данная модель представления знаний была предложена американским психологом Куиллианом. Основным ее преимуществом является то, что она более других соответствует современным представлениям об организации долговременной памяти человека

Недостатком этой модели является сложность организации процедуры поиска вывода на семантической сети.

19.1.2. Фреймы

Термин фрейм (от английского frame, что означает «каркас» или «рамка») был предложен *Марвином Минским*, одним из пионеров ИИ, в 70-е годы для обозначения структуры знаний для восприятия пространственных сцен. В то время еще существовал термин пакет ожидания. Эта модель, как и семантическая сеть, имеет глубокое психологическое обоснование.

Фрейм — это абстрактный образ для представления некоего стереотипа восприятия.

В психологии и философии известно понятие абстрактного образа. Например, произнесение вслух слова «комната» порождает у слушающих образ комнаты: «жилое помещение с четырьмя стенами, полом, потолком, Окнами и дверью, площадью 6-20 м^2». Из этого описания ничего нельзя убрать (например, убрав окна, мы получим уже чулан, а не комнату), но в нем есть

«дырки» или «слоты» — это незаполненные значения некоторых атрибутов — например, количество окон, цвет стен, высота потолка, покрытие пола и др.

В теории фреймов такой образ комнаты называется фреймом комнаты. Фреймом также называется и формализованная модель для отображения, образа.

Различают фреймы-образцы, или прототипы, хранящиеся в базе знаний, и фреймы-экземпляры, которые создаются для отображения реальных фактических ситуаций на основе поступающих данных. Модель фрейма является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний о мире через: фреймы-структуры, использующиеся для обозначения объектов и понятий (заем, залог, вексель);

- фреймы-роли (менеджер, кассир, клиент);
- фреймы-сценарии (банкротство, собрание акционеров, празднование именин);
- фреймы-ситуации (тревога, авария, рабочий режим устройства) и др.

Основным преимуществом фреймов как модели представления знаний является то, что она отражает концептуальную основу организации памяти человека а также ее гибкость и наглядность.

19.1.3. Правила вывода

Понятие *правила вывода* тесно связано с понятием машины вывода.

Машина вывода – программа, которая выполняет логический вывод из предварительно построенной базы фактов и правил в соответствии с законами формальной логики.

Под *фактом* понимается утверждение, являющееся постоянно истинным, например: «Василий является человеком».

Правило в данном случае — параметризованное утверждение, состоящее из двух частей: *условия и результата*, например: «Если X является потомком собаки, то X является собакой». В данном примере правило определяет условие «X является потомком собаки». Если для значения параметра X это условие истинно, то правило превращается в факт «X является собакой».

Базой знаний, в случае с машиной вывода, является база фактов и набор правил вывода.

20. Открытые системы и сетевые технологии

Открытость систем необходима для универсализации и взаимозаменяемости программно-аппаратных средств. Такая унификация может быть необходима при взаимодействии сетевых узлов, функционирующих на оборудовании разных производителей; при комплектации отдельных частей какой-то большой системы. Например, если

элементы системы являются интерфейсно-несовместимыми с оборудованием других поставщиков, то фирма, у которой закупались эти элементы, становится фактически монополистом и может диктовать свои условия при необходимости расширении системы (закупке дополнительного оборудования).

Примером прорыва в развитии открытых систем является производство персональных компьютеров, первые поколения которых были разработаны фирмой IBM. За счет открытости стандартов на интерфейсы и совместимости модулей различных производителей даже у небольших фирм появилась возможность косвенно участвовать в развитии «персонального компьютеростроения», занимая лишь небольшую нишу в общем производстве. В этом же ключе можно привести примеры с развитием технологий записи информации на лазерные диски и т.п.

Отдельной строкой в данном перечне стоят сетевые, телекоммуникационные технологии, призванные обеспечивать взаимодействие систем, принадлежащих разным компаниям, отдельным лицам, произведенные по уже устаревшим или самым современным технологиям.

У истоков унификации во взаимодействии открытых систем лежала разработка *семиуровневой модели OSI/ISO*. Однако, прежде чем рассматривать взаимодействие технических систем приведем простой эпизод из жизни. Так хорошим примером взаимодействия различных по структуре, но имеющих единый интерфейс систем может служить написание письма бельгийского бизнесмена арабскому. Бельгиец не знает арабского языка и пишет письмо своему коллеге на французском. Затем письмо попадает к переводчику который переводит письмо на английский язык. Англоязычное послание запечатывается в конверт и после проставления адреса отправителя и получателя посылается адресату. Почтовые службы доставляют письмо в офис арабскому бизнесмену, где работает переводчик с английского языка на арабский. Таким образом, послание в конце своего пути попадает к адресату на арабском языке.

Обратите внимание на то, что при прохождении корреспонденции в приведенном выше примере были задействованы две сущности, которые остались за кадром повествования, однако играют огромную роль как в рассмотренном примере, так и в реальной жизни, при передаче данных, функционировании сетей и т.п.

Этими сущностями являются *службы и протоколы.* Прежде чем двигаться дальше определим, что такое службы и что такое протоколы.

Протоколами мы называем определенный набор правил, которого придерживаются взаимодействующие объекты или субъекты. Обычно для примера приводят дипломатический протокол, который соблюдается на приеме послов или при визите глав государств.

Службами можно назвать те механизмы, которые осуществляют взаимодействие согласно протоколу.

В приведенном выше примере, в качестве одного из протоколов, можно выделить почтовый протокол, согласно которому заполняется почтовый адрес, а почтовую службу можно отнести к коммуникационному механизму, осуществляющему взаимодействие – пересылку письма адресату (только в том случае, если правильно заполнен адрес получателя и отправителя, т.е. соблюден коммуникационный протокол).

Детализация модели взаимодействия систем позволяет выделить следующие задачи:

- распознавание данных;
- разбиение данных на управляемые блоки;
- добавление информации к каждому блоку, чтобы указать местонахождение данных и указать получателя;
- добавление информации о синхронизации и информации для исправления ошибок;
- помещение данных в сеть и отправка их по заданному адресу.

В 1978 году International Standards Organisation (ISO) выпустила набор спецификаций, описывающих архитектуру сети с неоднородными устройствами. Исходный документ относился к открытым системам, чтобы они могли взаимодействовать, используя одинаковые стандарты и протоколы.

20.1. Семиуровневая модель OSI/ISO

В 1984 году выходит новая версия этого документа, которая носит название «*Эталонная модель взаимодействия открытых систем» Open System Interconnection (OSI).*

При создании спецификации на взаимодействие открытых систем весь коммуникационный процесс разбили на семь функционально различных этапов или уровней. По замыслу разработчиков, функциональная независимость каждого из этапов взаимодействия должна была позволить образовать отдельные группы программистов и инженеров, создающих совершенно самостоятельный продукт, совместимый только в плане интерфейсов с продуктами ближайших (нижнего и верхнего уровней). Все это позволило бы обеспечивать взаимодействие систем функционирующих на разных программно-аппаратных платформах, связанных различными коммуникационными каналами. Однако, как это бывает на практике, ни одна из реально работающих систем не декомпозируется на семь уровней в соответствии со ставшей уже классической семиуровневой моделью OSI/ISO. Но, тем не менее, эту модель приводят как эталонную для описания взаимодействия открытых систем.

Обычно уровни начинают перечислять сверху вниз по пути следования информации между конечными пользователями.

- Седьмой уровень называют уровнем приложений или прикладным уровнем;
- Шестой уровень – уровень представления;
- Пятый уровень – сеансовый уровень;
- Четвертый уровень – транспортный;
- Третий уровень – сетевой;
- Второй уровень – канальный;
- Первый уровень – физический.

Рассматривают два типа взаимодействия систем: вертикальное – между соседними уровнями и горизонтальное – между одноименными уровнями разных узлов.

Хотелось бы обратить ваше внимание на эффект инкапсуляции при прохождении информации с верхнего уровня на соседний нижний уровень. Этот эффект можно представить как обертывание пакетов вышестоящего уровня в служебную информацию текущего уровня.

И еще, взаимодействие построено таким образом, что каждый уровень предоставляет свои сервисы (услуги) соседнему вышестоящему уровню.

Рассмотрим задачи, которые выполняет *Прикладной уровень*.

Этот уровень является источником и приемником для всех передаваемых через сеть сообщений. Процессы прикладного уровня не всегда являются синонимами самих приложений. Например, текстовый процессор хочет открыть документ, сохраненный на сетевом диске. В текстовом процессоре нет процесса прикладного уровня, который бы обеспечивал доступ к сети. Эту задачу решает элемент сетевой операционной системы называемый редиректором, который умеет различать запросы, направленные к файлам, расположенным на локальном или сетевом дисках. Некоторые приложения специально разработаны для доступа к сетевым ресурсам. Например, приложение FTP.

Ниже перечислены протоколы, неразрывно связанные с приложениями их использующими:

DHCP (Dynamic Host Configuration Protocol, протокол динамической конфигурации хостов);

TFTP (Trivial File Transfer Protocol, простейший протокол передачи файлов);

DNS (Domain Name System, система доменных имен);

NFS (Network File System, сетевая файловая система);

OSPF (Open Shortest Path First, протокол маршрутизации с выбором кратчайшего маршрута).

Следующие протоколы функционируют как на Прикладном, так и на Представительском и Сеансовом уровнях:

SMTP (Simple Mail Transport Protocol, простой протокол пересылки почты);

POP3 (Post Office Protocol, почтовый протокол);

SNMP (Simple Network Management Protocol, простой протокол управления сетью);

HTTP (Hypertext Transfer Protocol, протокол передачи гипертекста);

NNTP (Network News Transfer Protocol, протокол передачи сетевых новостей);

FTP (File Transfer Protocol, протокол передачи файлов);

X.400 (протоколы и функции обработки сообщений электронной почты);

Telnet (эмуляция терминала, удаленное подключение к узлам сети);

Rlogin (удаленное подключение к узлам сети).

Уровень Представления данных (Presentation)

Этот уровень называется переводчиком. Данные переводятся в промежуточный формат и обратно (например, происходит преобразование кодовых таблиц для отображения на стандартных терминалах (vt100 и т.п.); соединение между PC и мэйнфреймом может потребовать перекодирования сообщений из ASCII в EBCDIC). На представительском уровне может происходить шифрование данных. На этом уровне работает утилита, называемая редиректором (redirector - программа, предназначенная для перехвата запросов OS ввода/вывода и перенаправления их к ресурсам сервера).

Сеансовый уровень. (session)

Позволяет двум приложениям на разных компьютерах устанавливать, использовать и завершать соединение. Сеансовый уровень обеспечивает синхронизацию между пользовательскими задачами посредством расстановки в наборе данных контрольных точек (check-points). Если в процессе взаимодействия произошла ошибка, то требуется повторить передачу не всех данных, а данных, следующих после очередной контрольной точки. На сеансовом уровне также происходит управление диалогом между взаимодействующими пользовательскими процессами, т.е. регулируется какая из сторон осуществляет передачу, когда и как долго. В Интернете данный уровень отсутствует, а его функции выполняются уровнем приложений.

Протоколы:

NetBIOS (Network Basic I/O System, сетевая базовая система ввода вывода)

NetBEUI (Network Extended User Interface, расширенная среда NetBIOS)

Данные протоколы частично захватывают также и 6 и 7 уровни.

Транспортный уровень

Обеспечивает дополнительный уровень соединения – ниже сеансового. ТУ гарантирует доставку пакетов без ошибок, в той же последовательности, без потерь и дублирования. На этом уровне сообщения переупаковываются: длинные сообщения разбиваются на несколько пакетов, а короткие объединяются в один. Повышается эффективность передачи пакетов в сети. На транспортном уровне получателя пакеты распаковываются, восстанавливаются в первоначальном виде и посылаются подтверждения. Все вышеперечисленные функции обеспечивает протокол *TCP (Transfer Control Protocol)* работающий с установкой соединения. Протокол *UDP (User Datagram Protocol)* работает без установки соединения. Этот протокол не посылает запросов на повторную передачу данных при возникновении ошибок. Используется для коротких транзакций (DNS), либо при передаче мультимедийной информации.

Функции транспортного протокола: сегментация и восстановление (пакеты); управление потоком данных (скорость передачи данных); контроль и устранение ошибок.

Протоколы:

TCP, UDP, SPX (Sequential Packet eXchange)

Три нижних уровня модели OSI/ISO называют подсетевыми (subnet). Службы этих уровней отвечают за доставку блоков данных от источника к приемнику. На этих уровнях работают различные коммутирующие устройства: маршрутизаторы, мосты, цепи X.25.

Сетевой уровень

Задачи сетевого уровня заключаются в определении адресов и преобразовании физических адресов в сетевые и обратно. Это могут быть как локальные адреса (MAC адреса) для локальных сетей, так и IP адреса для глобальных сетей.

При передаче данных между источником и приемником или между двумя промежуточными узлами в сети на сетевом уровне решается задача прокладки маршрута. При этом выполняются следующие действия:

- снимается оболочка канального уровня;
- проверяется целостность;
- проверяется адрес назначения сетевого уровня.

Если адрес удаленный, то в ход идет таблица маршрутизации; пакет передается канальному уровню и отправляется до ближайшего шлюза. Так как промежуточные системы могут иметь различные канальные уровни, то на

сетевом уровне датаграммы могут разбиваться на пакеты соответствующего размера.

Протоколы:

IP (Internet Protocol межсетевой протокол) работает с 32 разрядными IP адресами.

IPX (Internet Packet eXchange) работает с MAC адресами.

Функции: Маршрутизация, фрагментация, контроль ошибок, идентификация протокола транспортного уровня.

Можно использовать любой протокол канального уровня с любым протоколом сетевого уровня, однако сетевой протокол с протоколом транспортного уровня образует стек протоколов, который разрывать нельзя: (TCP/IP; IPX/SPX)

Канальный уровень

На канальном уровне осуществляется генерирование, передача и получение кадров данных. Протокол канального уровня обеспечивает интерфейс между физической сетью и стеком протоколов компьютера. Протокол канального уровня обычно состоит из трех элементов:

Кадра специального формата, который инкапсулирует данные протокола Сетевого уровня;

Механизма, регулирующего доступ к совместно используемой сетевой среде (CDMF/CD - Ethernet; с передачей маркера - Token Ring)

Принципов, которые должны быть реализованы при разработке Физического уровня сети.

Кадр канального уровня обычно содержит адресную информацию для доставки пакета по сети и средства обнаружения и исправления ошибок. Адрес кадра – MAC (Media Access Control) адрес представляет из себя число, состоящее из 6 байтов, три байта заполняются на заводе изготовителе. Алгоритм составления MAC адреса придуман таким образом, чтобы в мире не было ни одного сетевого адаптера с одинаковыми MAC адресами. Протоколы канального уровня не заботятся о доставке пакета конечному адресату, если получатель находится в той же ЛВС, что и отправитель. Если же конечный пользователь находится в другой локальной сети, то пакет адресуется маршрутизатору. Таким образом, адрес получателя в заголовке пакета Канального уровня всегда относится к устройству, расположенному в локальной сети. При обработке ошибок, если ошибка обнаружена, то пакет просто уничтожается без запроса на повторную передачу. Запрос на повторную передачу формируется на Транспортном уровне.

Кадры канального уровня делят на два типа:

пакеты (кадры данных);

маркеры (управляющие кадры).

В кадре Канального уровня предусмотрен индикатор протокола Сетевого уровня, с которым будет взаимодействовать данный пакет.

Протоколы канального уровня: SLIP и PPP.

Физический уровень

Осуществляет передачу неструктурированного (сырого) потока битов.

На Физическом уровне определяется способ соединения сетевого кабеля с платой сетевого адаптера. Уровень отвечает за кодирование данных и синхронизацию битов, гарантируя, что переданная единица будет принята как единица. Физический уровень устанавливает длительность каждого бита и способ перевода бита в соответствующий электрический или оптический импульс.

20.2. Передача данных по сети (функции пакетов)

Данные содержатся в больших по размеру файлах. Если передавать по сети файлы целиком, то сеть будет связана (занята) долгое время. Если передача файла будет прервана ошибкой, то повторная передача должна будет возобновиться с начала файла и также потребует длительного времени. Чтобы легче было передавать данные, необходимо разбить на управляемые блоки, которые называются пакетами или кадрами.

Пакет – основная единица информации в компьютерных сетях. Пакеты могут содержать несколько типов данных: информацию (например, сообщения или файлы); определенные виды данных и команд управления компьютером (например, запросы к службам); коды управления сеансом (например, запросы на повторную передачу недошедшего пакета).

20.3. Сетевое оборудование

Для обеспечения сетевого взаимодействия между компьютерами используется достаточно большое количество устройств сетевого оборудования.

Сетевой адаптер – устройство, отвечающее за связь компьютера с сетью (интерфейс между системной шиной компьютера и сетью). Функционирует на Канальном и Физическом уровнях. Сетевой адаптер осуществляет подготовку данных (преобразование из параллельного кода в последовательный и обратно), буферизацию, кодирование, гальваническую развязку; В настоящее время сетевые адаптеры встроены в материнскую плату компьютера.

Повторитель (reapeter) – полностью электронное устройство, которое увеличивает максимальную протяженность кабеля ЛВС путем усиления сигнала, проходящего через такое устройство. Концентраторы, используемые в

сетях, основанных на топологии «Звезда» иногда называются многопортовыми повторителями, поскольку сами по себе имеют способность к усилению сигнала.

Концентратор (Ethernet) – устройство, которое служит для объединения совокупности компьютеров в единую область коллизий. Существуют пассивные (устаревшие) и ретранслирующие (более современные) концентраторы. Также выпускаются интеллектуальные концентраторы, которые могут работать с различными портами персонально «отслеживая» работу каждого порта. Такие концентраторы могут работать под управлением протокола *SNMP (Simple Network Management Protocol).*

Точка доступа для беспроводных сетей (англ. Wireless Access Point, WAP) – это беспроводная базовая станция, предназначенная для обеспечения беспроводного доступа к уже существующей сети (беспроводной или проводной) или создания новой беспроводной сети. Следует учитывать, что точка доступа — это обычный концентратор. При нескольких подключениях к одной точке полоса пропускания делится на количество подключённых пользователей. Теоретически ограничений на количество подключений нет, но на практике стоит ограничиться, исходя из минимально необходимой скорости передачи данных для каждого пользователя.

Мосты (bridges) – устройства, которые выполняют функцию усиления сигнала, как и повторители, но вместе с тем имеет способность избирательно отфильтровывать пакеты по их адресам. Пакеты, приходящие на вход моста, пропускаются на выход только в том случае, если они адресованы компьютеру, находящемуся по другую сторону моста. Поскольку мосты не препятствуют прохождению широковещательных сообщений, они не делят ЛВС на сегменты и не создают интрасети.

Маршрутизация (routing) и коммутатация (switching) являются базовыми концепциями для построения корпоративных сетевых комплексов и формирования инфраструктуры самой большой интерсети – Интернета.

Коммутаторы – это устройства, которые во многих случаях устраняют необходимость среды передачи данных. Коммутатор является многопортовым повторителем, как и концентратор, однако, вместо работы на чисто электрическом уровне он считывает адрес назначения каждого входящего пакета и передает его только через тот порт, с которым соединен компьютер-адресат. Коммутаторы могут функционировать на разных уровнях, объединяя сети с другими сетями или сетевыми комплексами. Применение коммутаторов дает поразительный эффект повышения производительности. Существует два типа коммутаторов: без буферизации и с промежуточным хранением пакетов. Первый тип коммутаторов передает пакеты по назначению непосредственно по мере их поступления. Второй тип является более дорогостоящим, и может сохранять пакет некоторое время, если порт назначения занят.

Маршрутизатор – это устройство, соединяющее различные ЛВС и формирующее интерсеть. Равно как и мост, маршрутизатор пропускает только информацию, предназначенную для сегмента, с которым он соединен. Однако, в отличие от повторителей и мостов, маршрутизаторы препятствуют прохождению широковещательных сообщений. Они могут объединять сети различных типов (Ethernet и Token Ring), в то время как мосты и повторители могут интегрировать только однотипные сети или сетевые сегменты.

Основная функция маршрутизатора - обработать каждый пакет, полученный от одной из сетей, к которым он подключен, и передать пакет дальше в пункт его назначения через другую сеть (основной задачей является выбор оптимального маршрута). Каждый маршрутизатор на пути пакета обозначается как транзит (hop), конечная цель состоит в том, чтобы доставить пакет с наименьшим количеством транзитов. При выборе маршрута используется так называемая таблица маршрутизации. При выборе оптимального маршрута проводится анализ таблицы маршрутизации, который заключается в следующем. В начале протокол сетевого уровня (IP) сканирует таблицу маршрутизации в поисках маршрута к узлу, который в точности совпадает с адресом, содержащемся в пакете, если такой существует, то пакет передается шлюзу, указанному в записи таблицы. Если совпадающего адреса не существует, IP при помощи маски подсети определяет адрес сети и ищет в таблице маршрутизации записи, включающие этот адрес. Если совпадающее значение найдено, пакет передается либо соответствующему шлюзу (если система не подключена непосредственно к сети назначения), либо к отмеченному сетевому интерфейсу (если система -получатель находится в локальной сети). В случае, если в таблице маршрутизации нет заданного адреса сети, IP ищет маршрут по умолчанию (default или 0.0.0.0) и передает пакет указанному шлюзу. Если же таблица не содержит определения маршрута по умолчанию, IP возвращает отправителю пакета (либо приложению, создавшему пакет, либо системе, передавшей его) сообщение о том, что место назначения не достижимо.

Различия в подходах к решению проблемы маршрутизации заключается в способах обновления таблицы маршрутизации. Первый способ относится к статической маршрутизации и заключается в обновлении таблицы маршрутизации вручную (например, утилита route.exe). Второй способ - динамическая маршрутизация использует внешний протокол для сбора информации о сети (например, RIP - Routing Information Protocol).

20.4. Топологии локальных сетей

Раньше рассматривали три типа топологий локальных вычислительных сетей:

- Шина (все компьютеры имеют равный асинхронный доступ к сети);

- Кольцо (компьютеры соединены последовательно и имеют синхронный доступ к сети посредством маркера, при этом обеспечивается большая скорость передачи данных за счет отсутствия коллизий);
- Звезда (все компьютеры подключаются к сети через центральный хаб или концентратор).

Сейчас за счет широчайшей распространенности технологии Ethernet, поддерживающей топологию Звезда, остальные топологии встречаются достаточно редко.

20.5. Система адресации в сети Internet

Обмен данными между узлами в сети Интернет ведется с использованием IP-адресов. IP расшифровывается как Internet Protocol (Интернет протокол). Всякий IP-адрес состоит по меньшей мере из адреса сети и адреса узла в этой сети. Адрес сети занимает от одного до трех октетов (октет - это восьмерка битов, в общепринятой нотации представленная десятичным числом от 0 до 255 и отделенная от другого октета символом точки)

В Таблице 20.1 показаны три класса сетей и их характеристики

Таблица 20.1

Диапазон IP-адресов	Класс сети	Максимальное количество узлов сети	Максимальное количество сетей
001.xxx.xxx.xxx - 126.xxx.xxx.xxx	A	16777214	126
128.000.xxx.xxx - 191.255.xxx.xxx	B	65534	16382
192.000.000.xxx - 223.255.255.xxx	C	254	2097150

20.6. Доменная система имен

Сейчас сеть Интернет насчитывает несколько миллиардов компьютеров. Управлять такой сетью очень сложно. Для упрощения задачи управления, маршрутизации нахождения нужных адресов было решено по определенным признакам разделить всю сеть на подсети – домены. Основная часть доменов представлена по территориальному признаку, т.е. у каждой страны есть свой домен. Также существуют образовательные домены, правительственные домены, научные домены и т.п.

Вторым шагом к упрощению процессов управления глобальной сетью явилось введение доменных имен. Т.е. вместо того, чтобы обращаться к компьютеру по адресу пользователь может просто набрать осмысленное слово (или набор слов).

Третий шаг к упрощению процессов управления глобальной сетью явилось создание иерархии доменов. Системный анализ показывает, что для упрощения понимания и управления любой, и не только технически сложной, системой может послужить иерархия связей между элементами системы. Иерархия позволяет разбить весь процесс управления на подпроцессы, управляющие только своей подчиненной частью, причем разбить единообразно для каждого уровня иерархии. Соответственно и имя каждого узла сети формируется из иерархии имен вышестоящих доменов. Если ваш компьютер имеет имя *robionica* и зарегистрирован непосредственно в домене страны (ru), то обратиться к нему можно *robionica.ru* (в определенных случаях www.robionica.ru).

Теперь скажем несколько слов о механизмах функционирования доменной системы имен глобальной сети Интернет. Когда пользователь набирает доменный (буквенный адрес) нужного ему узла (а маршрутизатору необходим цифровой адрес), то происходит операция разыменования, т.е. получения адреса по имени. Фактически происходит обращение к серверу имен текущего домена (DNS-серверу). Если сервер доменных имен не знает адреса для нужного вам имени, то он обращается к серверу имен вышестоящего домена. Корневой домен при таком поиске может перенаправить запрос к серверу имен нужного домена. На первый взгляд такой иерархический подход кажется достаточно длинным, однако большинство адресов, к которым обращаются пользователи хранятся в КЭШ памяти доменного сервера имен вашего уровня (либо вообще на вашем компьютере). Отметим, что вы можете обращаться к любому узлу по цифровому адресу, минуя доменную систему имен, но это не очень удобно.

20.7. Порты и сокеты

Если обмен происходит на уровне приложений, запущенных на разных компьютерах, то одного IP-адреса или доменного имени оказывается мало (на компьютере может быть запущено несколько приложений, требующих «персональной» адресации). Для решения подобных задач каждому приложению (особенно из числа часто используемых) договорились присваивать цифровые номера, называемые портами (port). *Порт* — цифровой номер, который является программным адресом, используемым для взаимодействия различных конечных точек.

В этом же контексте появилось и стало использоваться понятие *сокет (socket) – порт + IP-адрес*. Например, если мы хотим обратится к WEB-серверу (myServ.ru) по стандартному HTTP протоколу, то в строке адреса мы должны набрать адрес сокета: *http://myServ.ru:8080* (во многих браузерах он добавляется по умолчанию). Для обеспечения в некотором смысле «скрытости» соединения можно пользоваться не стандартными портами, например настраивать WEB-сервер только «для своих» на порт 8088 и т.п.

Отметим, что в некоторых источниках сокетом называют просто программный интерфейс (что сути не меняет и не противоречит нашему определению).

21. Информационная безопасность

Исторический опыт показывает, что информация попавшая в руки противника во время боевых действий может нанести урон гораздо больший, чем несколько дивизий с живой силой и техникой. Например, зная время, место наступления и наступающие силы, можно своевременно подготовиться, подтянуть резервы, перегруппировать войска и нанести существенный урон противнику. С другой стороны, гораздо проще наступать, если известно расположение фортификационных сооружений, численный состав обороняющихся войск и их вооружение. Зная имена командного состава, достаточно просто дезорганизовать оборону противника, передавая ложные приказы от имени вышестоящих командиров. Как пример, можно привести начало Великой Отечественной войны, когда многочисленные диверсионные группы немцев под видом красноармейцев проникали в тыл советских войск, перерезали линии связи, сеяли панику и выводили из строя и без того ненадежную систему управления войсками.

Другой пример – чеченская война, середина 90-х годов. Использование нашими войсками устаревшей аппаратуры и незащищенных каналов связи, большое скопление подразделений разного подчинения, отсутствие продуманности и порядка в управлении войсками позволяло боевикам прослушивать радиопереговоры, получая информацию о расположении и передвижении войск, а также ложными сообщениями от имени известных позывных корректировать огонь артиллерии, так, чтобы он накрывал наши подразделения.

В мирное время, зная маршруты и расписания перемещений материальных ценностей или субъекта, на которого планируется покушение, фиксируя связанные с объектом радио и телефонные переговоры, можно подготовить засаду и эффективно выполнить задуманную операцию.

Даже если не рассматривать несанкционированное использование информации во время силовых операций, то можно привести много примеров того, как утечка информации способна послужить причиной большого количества проблем. В качестве одного из примеров такого рода можно привести промышленный шпионаж. Известно, что при разработке новых изделий или технологий тратятся огромные деньги на проведение исследований и эксперименты, доведение изделий до серийного образца. Если результаты таких исследований попадают в руки конкурирующей компании, то она экономит огромное количество финансовых и человеческих ресурсов и получает результат практически даром, оставляя в убытке реального разработчика.

Еще один пример – проведение тендеров и аукционов. Если заранее известно, кто и за какие деньги готов приобрести тот или иной выставляемый на торги объект или проект, то нетрудно выстроить свою стратегию ведения переговоров и ценовую политику. А ведь речь идет порой о многомиллионных контрактах.

Надежность и защищенность платежных систем – это вообще отдельная тема для разговора. Возможность взлома электронных платежей киберпреступниками может нанести не только финансовый урон кредитной организации, но и навредить репутации того или иного банка или группы банков.

Также как и утечка информации, опасна дезинформация. Например, в конце тридцатых годов прошлого века дезинформация о причастности высокопоставленных руководителей Советского Союза к шпионажу или просто ложные доносы, служили причиной их арестов, и, как следствие, влекли нарушение функционирования целых отраслей промышленности.

И в наше время умело подобранная информация, выложенные в Интернет высказывания от чьего-то имени, сгенерированные аудио и фотоматериалы могут быть выставлены в качестве компромата и быть причиной отставки высокопоставленных лиц со своих постов.

В университетской среде проблемы информационной безопасности, особенно на Западе, связаны с несанкционированным доступом к содержанию контрольных и тестовых заданий и ответов к ним. Также существуют проблемы с исправлением оценок в электронных журналах, с подменой или порчей персональных страничек на сайтах и в социальных сетях и т.п.

Надо сказать, что проблемы информационной безопасности наряду с большими и мелкими компаниями не обходят стороной и простых пользователей. В основном они связаны со спамом, вирусами, взломом электронной почты, страничек в социальных сетях.

Жизнь показывает, что, если существуют угрозы, то есть и способы защиты от них. Обычно информационная безопасность как страны, компании так и отдельного человека строиться на умело применяемых правовых, организационных и технических средствах и методах защиты.

21.1. Правовые меры защиты информации

Остановимся сначала на правовых мерах. Один из способов защиты – это доведение информации до потенциальных преступников и нарушителей о возможных последствиях и наказаниях за производимые ими действия.

Существует Федеральный закон № 149-ФЗ "Об информации, информационных технологиях и о защите информации", принятый 27 июля

2006 г. Например статья 16 этого закона рассматривает основные понятия и положения, касающиеся защиты информации:

• пункт 1: «Защита информации представляет собой принятие правовых, организационных и технических мер, направленных на:

1) обеспечение защиты информации от неправомерного доступа, уничтожения, модифицирования, блокирования, копирования, предоставления, распространения, а также от иных неправомерных действий в отношении такой информации;

2) соблюдение конфиденциальности информации ограниченного доступа,

3) реализацию права на доступ к информации»;

Еще один законодательный акт – Федеральный закон № 152-ФЗ «О персональных данных», вступивший в действие 27 июля 2006 г., определяет основные понятия и положения о защите персональной, т.е. личной информации каждого человека.

Какие наказания грозят тем, кто создает вредоносные программы или производит какие-то вредоносные действия, либо пытается получить несанкционированный доступ к чужой информации?

В России соответствующие меры определены в Уголовном кодексе Российской Федерации (УК РФ), носящем номер 63-ФЗ и вступившем в действие 13.06.1996 г. Преступления, связанные с компьютерной информацией, рассмотрены в нем в главе 28, где содержатся три следующие статьи:

• статья 272. Неправомерный доступ к компьютерной информации - наказывается штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев, либо исправительными работами на срок от шести месяцев до одного года, либо лишением свободы на срок до двух лет.

• статья 273. Создание, использование и распространение вредоносных программ для ЭВМ - наказываются лишением свободы на срок до трех лет со штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев.

• статья 274. Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети наказывается лишением права занимать определенные должности или заниматься определенной деятельностью на срок до пяти лет, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо ограничением свободы на срок до двух лет.

Ответчиков по компьютерным преступлениям обычно делят на три категории: пираты, хакеры, крекеры (взломщики).

- Пираты, главным образом, нарушают авторское право;

- Хакеры незаконно получают доступ к компьютерам других пользователей;
- Крекеры «взламывают» банковские системы, совершают кражи, шантажируют частные и государственные организации.

Отметим, что за рубежом выделяют еще и снифферов, которые осуществляют незаконный и бесплатный доступ к телефонным сетям.

Существует также ряд указов, постановлений и приказов, регламентирующих деятельность юридических лиц в области криптографии.

21.2. Организационные меры защиты информации

Что же делать простым пользователям, чтобы справиться с информационными угрозами и хотя бы в какой-то мере обезопасить себя от плачевных последствий «хакерских» или вирусных атак на свой компьютер. Ответ достаточно простой, и похож на совет «мыть руки перед едой».

Во-первых, не стоит «исследовать» сайты с порнографическим содержанием, обычно они полны различных вирусов.

Во-вторых, не загружать и не запускать программы из непроверенных источников.

В-третьих, не работать на компьютере под учетной записью администратора. Данный совет выполнить достаточно просто, так как современные операционные системы (Windows-7, Windows-8) запускают приложения с правами обычных пользователей и сообщают, если программа хочет выполнить какое-то опасное действие с другими правами.

В-четвертых, отключить автозапуск программ с CD, DVD, и Flash накопителей, хотя это и не очень удобно. Следить за появление непонятных «спрятанных» от просмотра файлов на Flash накопителе.

В-пятых, стараться периодически архивировать и сохранять нужные документы на внешнем носителе информации, так как вирусу достаточно сложно получить доступ к исполняемым файлам на компьютере под управлением современной ОС, а к файлу документа, чтобы его испортить – легко.

В-шестых, не открывать письма от незнакомых источников, особенно с прикрепленными фалами.

В-седьмых, работать в браузере с отключенным выполнением скриптов (это также не очень удобно).

В-восьмых, периодически чистить кэш браузера и куки (хотя это занимает определенное время).

В-девятых, никогда не соглашаться на сохранение паролей при входе на какой-то сетевой ресурс (так как очень часто пароли сохраняются в «открытом» виде).

В-десятых, для задания пароля своего аккаунта использовать длинные и трудно подбираемые слова.

В-одиннадцатых, стараться не пользоваться взломанным ПО, так как зачатую такие программы содержат вирусы.

В-двенадцатых, в случае необходимости применять повышенные меры защиты, можно отключать серверные службы вашей ОС. (хотя это приносит некоторые неудобства)

В-тринадцатых, стараться как можно меньше использовать на своем компьютере отдаваемые в общее пользование (Share) ресурсы;

В-четырнадцатых, периодически изучать сообщения в системном журнале и журнале безопасности своего компьютера.

В-пятнадцатых, при покупке какого-то компьютеризированного устройства, смартфона, мини-АТС, сетевого коммутатора и т.п., с функциями Remote Controle (удаленного управления) необходимо поменять пароль по умолчанию на свой собственный.

Если пользователь имеет желание или необходимость более серьезно отнестись к вопросам защиты информации, например, при работе с государственной тайной, то он вынужден использовать более серьезные организационные мероприятия, которые зависят от степени секретности и важности информации. Степень секретности сведений, составляющих государственную тайну, соответствует степени тяжести ущерба, который может быть нанесён государственной безопасности вследствие распространения указанных сведений. Устанавливаются три степени секретности сведений, составляющих государственную тайну, и соответствующие этим степеням грифы секретности: «особой важности», «совершенно секретно» и «секретно». Использование перечисленных грифов секретности для засекречивания сведений, не отнесенных к государственной тайне, не допускается.

Существует еще одна степень закрытости информации, так называемая информация «для служебного пользования» или конфиденциальная информация. Этот тип информации вводится по усмотрения предприятия или организации и процессы ее обработки и хранения также регламентируются на локальном уровне.

В зависимости от степени секретности информации определяется порядок работы с этой информацией: порядок хранения, перечень допущенных лиц, порядок выдачи и возврата этой информации, порядок копирования информации. Обязательно указывается количество экземпляров существования данной информации. Существует также регламент «вывода из эксплуатации»

секретной информации, начиная от способа уничтожения, заканчивая перечнем лиц в комиссии, отвечающих за соблюдение всех правил.

При работе с секретной информацией соблюдаются достаточно жесткие правила работы с техническими средствами ее обработки и хранения.

Компьютеры для обработки такой информации являются специализированными и проверенными, используются только в локальном варианте доступа (без каких либо сетевых соединений). Физический доступ к таким компьютерам и к сопредельным помещениям строго ограничен. Работать на таких компьютерах можно только в закрытом (лучше экранированном помещении) далеко от окон.

21.3. Технические средства защиты информации

Помимо правовых и организационных мер защищать информацию помогают конечно же и самые разнообразные технические средства.

Например, в помещениях, где происходят важные переговоры, используют тяжелые звукопоглощающие шторы и рефленые стекла, отражающие свет под разными углами, уменьшая, тем самым вероятность использования лазерных считывателей информации с обыкновенных стекол. Также используются низко, высокочастотные и широкополосные «глушилки» как звукового, так и радиодиапазона, исключая, тем самым, возможность использования обыкновенных и радио микрофонов.

Для закрытия информации «от посторонних ушей» при передаче данных по радиоканалу также используются сигналы с «прыгающей» несущей частотой. В последнее время появилась возможность использования шумоподобных сигналов (спектр, которых практически не различим на фоне общего шума). Также могут быть использованы очень короткие по времени сеансы связи, осуществляемые, например, через воду водоемов (чтобы труднее было засечь наличие сигнала).

Особенно часто при передачи информации через Интернет используют методы стеганографии (сокрытия самого факта передачи информации). Когда кому-то отправляется открытка или фотография местности, а в определенном углу в графическом векторном редакторе при большом увеличении нанесена надпись. Сообщение также может быть просто "проксорено" (Xor сумма по модулю 2) с данными изображения и тем самым просто «растворено» в нем.

Для предотвращения снятия информации с экрана компьютера на расстоянии (до 100 метров) в виде промышленного излучения, компьютеры обрабатывают специальным экранирующим составом, а сетевое питание фильтруют специальными устройствами.

Для уменьшения вероятности перехвата телефонных и радио переговоров используют скремблирование (искажение) аналогового речевого сигнала при

передаче его в канал связи с последующим восстановлением на приемной стороне.

Поскольку подавляющее большинство современных средств связи работают с цифровой информацией, то для обеспечения конфиденциальности используют шифрование. О шифровании и в целом о криптографии можно говорить очень много, начиная от истории развития, заканчивая современными программно-аппаратными комплексами. Единственное на что хотелось бы обратить внимание, так это два момента, о которых никогда не стоит забывать:

- Каким бы математически стойким не был применяемый алгоритм шифрования, в конечном итоге, результат его применения зависит от программно-аппаратной или чисто программной реализации. Во-первых, эта реализация может быть выполнена просто некачественно и с ошибками, а во-вторых, разработчики могут по требованию службы сертификации специально «оставить» возможность использования «универсального» пароля для чтения спецслужбами любых сообщений, зашифрованных данным ПО (Back Door)
- Также надо помнить о человеческом факторе, который путем подкупа или в результате шантажа может оказаться самым слабым звеном в любой технически-совершенной системе защиты информации.

Используя шифрование для защиты информации, следует понимать, что очень серьезную роль в этом вопросе играет реализация «правильной» политики обмена ключевой информацией между всеми источниками криптообмена.

Для осуществления платежей в системах Интернет банкинга используют систему разовых паролей или SMS паролей. Также для обеспечения надежности используют SSL криптографический протокол. SSL (англ. Secure Sockets Layer — уровень защищённых сокетов). Этот протокол использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений Протокол SSL позволяет общаться клиенту с сервером в сети, предотвращая перехват или фальсификацию.

Парольная защита самого компьютера обычно бывает эффективной только при использовании специальных устройств, предотвращающих прерывание процесса загрузки операционной системы (самое уязвимое место, когда можно получить доступ к диску компьютера).

Наряду с шифрованием необходимо отметить и электронную подпись (программную реализацию специальной «односторонней» функции), которая служит для аутентификации пользователей, как при выполнении электронных платежей, так и при удостоверении подлинности электронных документов, например, при предоставлении отчетности через Интернет.

Здесь же можно сказать о различных электронных ключах и сертификатах, служащих против несанкционированного пиратского использования программного обеспечения.

Также отдельной строкой необходимо отметить специальные средства уничтожения информации как на электронных, так и на бумажных носителях. Например, очевидно, что для удаления файла совершенно не достаточно просто «удалить» файл, нажав клавишу «Delete». При выполнении этой операции в имя файла, которое располагается в таблице размещения файлов, ставится специальная метка, а сама информация остается нетронутой до тех пор пока не будет занята другими файлами. Такой файл не составляет труда восстановить. Для более «полноценного» удаления файла пользуются программами, работающими по принципу дефрагментаторов диска. Но и они не дают стопроцентной гарантии уничтожения. Известны программно-аппаратные устройства, частично восстанавливающие многократно перезаписанную информацию по остаточной намагниченности носителя. Для стопроцентного удаления информации с дисков некоторое время назад использовались специальные «размагничиватели» дисков. Для уничтожения бумажных носителей информации используются так называемые шредеры— устройства, режущие бумагу на мелкие полосы, которые впоследствии перемешиваются.

Из технических (точнее из программных) средств защиты информации, с антивирусами простой пользователь сталкивается чаще всего. Какое-то предпочтение тому или иному антивирусу отдать сложно. В основном все они сравнивают части проверяемых файлов с сигнатурами известных вирусов (набор которых пополняется при обновлениях) и сигнализируют, если совпадение произошло. При прочих равных условиях, на наш взгляд, лучше использовать антивирус производителя операционной системы, хотя во многих случаях данное правило не работает. Антивирусы могут быть интегрированы с антиспамерами, фильтрами, анти шпионами и т.п.

Следующие по распространенности – это адресные фильтры–атиспамеры почтовых клиентов, блокирующие прием корреспонденции с определенных адресов.

Еще одной весьма полезной во многих отношениях программой является *Прокси (proxy) сервер*. Это служба или комплекс программ, которые применяются для:

- Обеспечения доступа компьютеров локальной сети к сети Интернет;
- Кэширования данных: если часто происходят обращения к одним и тем же внешним ресурсам;
- Защиты локальной сети от внешнего доступа: например, можно настроить прокси-сервер так, что локальные компьютеры будут обращаться к внешним ресурсам только через него, а внешние

компьютеры не смогут обращаться к локальным вообще (они «видят» только прокси-сервер);

- Ограничения доступа из локальной сети к внешней: например, можно запретить доступ к определённым web-сайтам ограничить использование интернета каким-то локальным пользователям, устанавливать квоты на трафик или полосу пропускания, фильтровать рекламу и вирусы.

Похожую функцию выполняет программа, которая носит название FireWall или Брандмауэр (английское и немецкое название пожарной огнезащитной стены). FireWall является разновидностью межсетевого экрана, может исключать из взаимодействия по сети определенные адреса, вести журнал безопасности и выполнять многие другие функции.

По поводу большинства средств защиты, которые мы рассмотрели выше, можно сказать одно: для эффективного использования всех программно-аппаратных средств, необходимо производить их грамотную настройку и конфигурирование. Настройки по умолчанию обычно желаемого эффекта не дают.

Во многих в учебных пособиях по информатике достаточно большая часть текста отводится для «оконного» описания большого количества «современных» (времени написания книги) прикладных программ. Однако, опыт показывает, что программное обеспечение развивается настолько быстро, что книги которые посвящены описанию картинок (фотографий экранов) различных прикладных программ устаревают настолько быстро, что через один - два года после выхода тиража, такая книга теряет свою актуальность. Поэтому в данном курсе на описании прикладных программ мы останавливаться не будем.

Для закрепления представленного в данном пособии материала в заключение хотелось бы привести набор контрольных и проверочных вопросов по некоторым разделам, которые были рассмотрены выше.

22. Приложение 1. Задачи по измерению количества информации

1. Алфавит роботов Zuko состоит из 32 букв. Какое количество информации «несет» одна буква этого алфавита?

2. Сообщение о том, что ваш друг живет на 8 этаже, кодируется пятью битами информации. Сколько этажей в доме?

3. Найти максимальное количество книг (каждая объемом 200 страниц, на каждой странице 60 строк, 80 символов в строке), полностью размещенных на лазерном диске емкостью 600 Мбайт.

4. Чему равно количество бит информации в сообщении «Федор на олимпиаде по информатике занял одно из 8 мест»?

5. Книга, набранная с помощью компьютера, содержит 150 страниц; на каждой странице - 40 строк, в каждой строке - 60 символов. Каков объем информации в книге? Определить в Кбайтах.

6. При угадывании целого числа в диапазоне от 1 до N было получено 8 бит информации. Чему равно N?

7. Сообщение, записанное буквами из 32-х символьного алфавита, содержит 30 символов. Какой объем памяти нужен для хранения этого сообщения?

8. «В гараже ли твой автомобиль?» – спросили водителя. «Нет», – ответил он. Сколько информации содержит ответ?

9. Сообщение занимает 3 страницы по 25 строк. В каждой строке записано по 60 символов. Сколько символов в использованном алфавите, если все сообщение содержит 1125 байт?

10. В коробке лежат 16 разноцветных шаров. Какое количество информации содержит сообщение, что из коробки достали желтый шар?

11. При угадывании целого числа в некотором диапазоне было получено 5 бит информации. Сколько чисел содержит этот диапазон?

12. Цветное (с палитрой из 256 цветов) растровое графическое изображение имеет размер 10x10 точек. Какой объем памяти займет это изображение?

13. Какое количество информации получит второй игрок после первого хода первого игрока в игре «крестики-нолики» на поле 4x4?

23. Приложение 2. Набор карточек для самостоятельной работы

Вариант 1

- Как отличить звуковой файл от файла, содержащего исполняемую программу
- Чем отличается транслятор от интерпретатора
- Какое программное обеспечение требуется для разработки других программ.
- Перечислить типы данных, которые используются для объявления целочисленных переменных.
- В каком из двух случаев требуется больше памяти для хранения объявленных переменных: a) dim mm as long, yy as double b) dim xx as integer, yy as single, zz as byte;
- Сколько байт памяти будет зарезервировано при объявлении переменных в пункте a) предыдущей задачи
- Двоичное число 11100101 представить в десятичном, восьмеричном и шестнадцатеричном видах.
- Десятичное число 33 представить в двоичном и восьмеричном видах.

Вариант 2

- Как отличить текстовый файл от файла, содержащего графическую информацию
- Чем отличается транслятор от дебаггера
- Какое программное обеспечение требуется для разработки других программ.
- Перечислить типы данных, которые используются для объявления строковых переменных.
- В каком из двух случаев требуется больше памяти для хранения объявленных переменных: a) dim x as byte, y as double b) dim x as integer, y as single, z as byte;
- Сколько байт памяти будет зарезервировано при объявлении переменных в пункте b) предыдущей задачи
- Двоичное число 11011100 представить в десятичном, восьмеричном и шестнадцатеричном видах.
- Десятичное число 101 представить в двоичном и восьмеричном видах.

Вариант 3

- Как отличить текстовый файл от файла, содержащего исполняемую программу
- Чем отличается компилятор от интерпретатора
- Какое программное обеспечение требуется для разработки других программ.
- Перечислить типы данных, которые используются для объявления целочисленных переменных.
- В каком из двух случаев требуется больше памяти для хранения объявленных переменных: a) dim x as long, y as double b) dim x as integer, y as single, z as byte, zz as byte;
- Сколько байт памяти будет зарезервировано при объявлении переменных в пункте b) предыдущей задачи
- Двоичное число 11010101 представить в десятичном, восьмеричном и шестнадцатеричном видах.
- Десятичное число 51 представить в двоичном и восьмеричном видах.

Вариант 4

- Как отличить звуковой файл от файла, содержащего исполняемую программу
- Чем отличается транслятор от компилятора
- Какое программное обеспечение требуется для разработки других программ.
- Перечислить типы данных, которые используются для объявления целочисленных переменных.
- В каком из двух случаев требуется больше памяти для хранения объявленных переменных: a) dim x as double, y as single b) dim x as integer, y as integer, z as long;
- Сколько байт памяти будет зарезервировано при объявлении переменных в пункте b) предыдущей задачи
- Двоичное число 11011100 представить в десятичном, восьмеричном и шестнадцатеричном видах.
- Десятичное число 37 представить в двоичном и восьмеричном видах.

Вариант 5

- Как отличить текстовый файл от файла, содержащего исполняемую программу
- Чем отличается транслятор от интерпретатора
- Какие программное обеспечение требуется для разработки других программ.
- Перечислить типы данных, которые используются для объявления целочисленных переменных.
- В каком из двух случаев требуется больше памяти для хранения объявленных переменных: a) dim x as long, y as double b) dim x as integer, y as single, z as long;
- Сколько байт памяти будет зарезервировано при объявлении переменных в пункте b) предыдущей задачи
- Двоичное число 11111100 представить в десятичном, восьмеричном и шестнадцатеричном видах.
- Десятичное число 13 представить в двоичном и восьмеричном видах.

24. Приложение 3. Контрольные проверочные вопросы

1. Информация в окружающем нас мире. Различные определения информации. Наука информатика. История развития информатики и ее связь с кибернетикой.

3. Операции сложения и умножения в двоичной системе счисления. Модули вычислительной системы, реализующие данные операции.

4. Управляющая информация хранится в четырех двухбайтовых регистрах. Сколько триггеров необходимо использовать для хранения такого же объема информации?

5. Способы измерения количества информации. Вычисление количества информации при равновероятных исходах событий. Формула Хартли. Биты, байты, килобайты, мегабайты, гигабайты, терабайты, рассказать о взаимосвязи перечисленных выше единиц измерения информации.

6. Теория трехкомпонентного цвета. Представление изображений в компьютере. RGB. CMYK. Способ вычисления объема памяти, необходимой для хранения изображения заданного размера (разрешения) и заданного качества (количества бит на цвет). Растровые и векторные изображения. Редакторы растровых и векторных изображений.

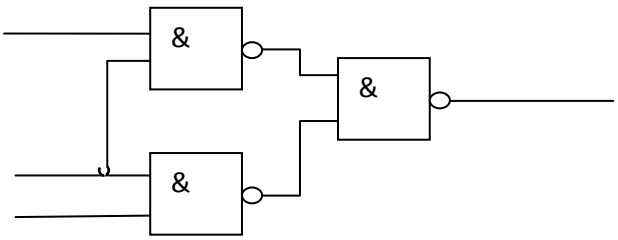
7. Привести таблицу истинности, соответствующую следующей формуле:

$$X_1 \& X_2 \vee \overline{X_1}$$

8. Преобразование чисел из позиционной системы счисления с одним основанием в позиционную систему с другим основанием. Объяснить необходимость таких преобразований и различные подходы к их осуществлению.

9. Аппаратные основы вычислительной техники. RS-триггеры, D-триггеры, T-триггеры. Структура RS-триггера. Двоичные счетчики. Назначение и составные элементы счетчиков. Регистры. Параллельные и сдвиговые регистры.

10. Дана логическая схема. Необходимо составить таблицу истинности, соответствующую данной схеме.



11. Представление звуковой информации в компьютере. Что такое ЦАП и АЦП. Зависимость качества представления звука и объема звукового файла от частоты дискретизации и от точности квантования уровня сигнала (разрядности преобразования).

12. Способы представления информации. Модели окружающего мира. Математические модели. Адекватность модели.

13. Для задания цвета одного пикселя при кодировании по схеме RGB используется три байта. Какое количество цветовых оттенков можно задать таким способом.

14. Способы создания, преобразования, хранения и передачи информации. Сигналы. Кодирование информации. Привести примеры различных кодировок текстовой информации в компьютере.

15. Логические основы вычислительных систем. Элементарные булевы функции (все функции от одной переменной и семь функций от двух переменных). Привести примеры таблиц истинности и обозначений вентилей в схемах, реализующих логические функции.

16. Посчитать пропускную способность шины AGP в Гбайт/с, если известно, что шина имеет разрядность 4 байта и работает с частотой 528 МГц

17. Системы счисления. Позиционные системы с различными основаниями (2, 8, 10, 16). Привести примеры представления одного и того же числа в этих четырех системах счисления. Пояснить необходимость знания для программиста систем счисления с основаниями 2, 8, 16.

18. Полный двоичный сумматор. В каком модуле вычислительной системы используется. Задать таблицу истинности одноразрядного полного двоичного сумматора. Записать СДНФ для функции суммы и для функции переноса в старший разряд для полного одноразрядного двоичного сумматора.

19. Сколько секунд фрагмента музыкального произведения содержится в звуковом файле размером 1Мбайт, если известно, что запись осуществлялась простым аналогово-цифровым преобразованием без сжатия с битрейтом 16 КГц, и количеством уровней квантования амплитуды сигнала равным 1024

20. Информация о событиях случающихся с различной вероятностью. Энтропия. Формула Шеннона. Связь энтропии с теорией кодирования сообщений.

21. Дополнительный код двоичного числа. Операция вычитания в двоичной системе счисления. Описать устройство вычислительной системы, которое осуществляет операцию вычитания.

22. Вычислить чему будет равна сумма двух восьмеричных чисел 17 и 34, представленная в десятичной и шестнадцатеричной системах счисления.

23. Представление булевой функции в виде формулы. Конъюнкции и дизъюнкции. Что такое ранг конъюнкции и ранг дизъюнкции. Упрощение формул булевых функций. Формулы эквивалентных преобразований.

24. Простейшая модель вычислительной системы. Привести пример структуры вычислительной системы, соответствующей данной модели и основные шаги функционирования такой системы.

25. Вычислить сколько информации содержится в сообщении о том что в группе, состоящей из двух девушек и тридцати юношей, победителем олимпиады стал юноша.

26. Модель современной вычислительной системы (персонального компьютера) с шинной (магистральной) архитектурой. Перечислить основные модули вычислительной системы: Процессор, память... и типы шин (магистралей), посредством которых происходит взаимодействие между модулями.

27. Из приведенных ниже формул выбрать тождественно истинную:

1) $A \& \overline{A} \vee \overline{A}$ 2) $A \& \overline{B} \vee A$ 3) $A \& \overline{A} \vee \overline{B}$ 4) $A \vee \overline{B} \vee \overline{A}$

28. Базовые элементы построения вычислительной системы: мультиплексоры, демультиплексоры, дешифраторы. Простой двоичный дешифратор. Дешифратор семисегментного индикатора; привести пример таблицы истинности, описывающей функционирование дешифратора семисегментного индикатора.

29. Определить какое из названий перечисленных ниже устройств не удовлетворяет следующему правилу:

(первая согласная -> вторая согласная) & (предпоследняя согласная->последняя гласная)

Процессор; Мультиплексор; Аккумулятор; Ионистор

30. Программное обеспечение вычислительной системы. История развития программного обеспечения. Основные типы программного обеспечения и их роль в работе вычислительной системы (перечислить и привести примеры).

31. История развития компьютеров во второй половине XX века. Классификация компьютеров по используемой для их создания элементной базе. Поколения компьютеров. Закон Мура.

32. Указать какое из перечисленных ниже чисел записано в восьмеричной системе счисления: 105987, 193, 7345, 2850, A114, FF07

33. Операционные системы. История возникновения и развития операционных систем. Перечислить основные семейства операционных систем. Описать базовые элементы структуры операционной системы. Описать последовательность загрузки компьютера.

34. Дать определение архитектуры вычислительной системы. Принципы построения вычислительных систем, описанные фон Нейманом (история формулировки принципов фон Неймана). Гарвардская и Принстонская архитектуры вычислительной системы (преимущества и недостатки).

35. В книге 200 страниц. На каждой странице по 40 строк, каждая строка содержит по 65 символов. Сколько байт, килобайт и мегабайт понадобится для хранения данной книги в ASCII кодировке.

36. Назначение и архитектура процессора вычислительной системы. Микропроцессоры и микроконтроллеры (история развития и современный уровень). Технология производства микропроцессоров. Закон Мура. Дать определение КЭШ памяти. Сколько уровней КЭШ используется в современном процессоре? Системы команд процессора. RISC и CISC архитектуры.

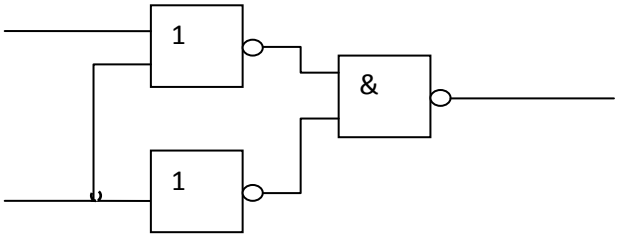
37. Локальные и глобальные компьютерные сети. Сетевая архитектура. Сетевая топология. Способы адресации в сетях. IP-адрес, маска сети. Домены. DNS. Сетевое оборудование и сетевое ПО.

38. В лесопарковой зоне, расположенной недалеко от города, живет 2400 кроликов и 600 зайцев. Определить сколько бит информации содержится в рассказе детей о том что они во время прогулки видели в лесу одного зайца.

39. Протоколы сетевых взаимодействий. Привести примеры сетевых протоколов функционирующих на различных уровнях семиуровневой модели OSI.

40. История развития вычислительных устройств до середины 20 века. Изобретатели XVII века. Изобретатели IX века. Электромеханические и аналоговые компьютеры первой половины XX века. История использования перфокарт, история фирмы IBM.

41. Записать таблицу истинности булевой функции, реализуемой следующей схемой:



42. Компьютерная безопасность. Правовая защита. Шифрование данных. Вирусы и антивирусные средства. Хакеры. Методы защиты от хакерских атак.

43. Память компьютера. RAM. ROM. Технологии изготовления различных типов полупроводниковых запоминающих устройств: FLASH. Статическая, Динамическая память.

44. Ученик за два года получил 100 оценок: 50 пятерок, 30 четверок, 15 троек, 5 двоек. Определить сколько информации содержится в сообщении о том, что ученик получил четверку.

45. История развития мировой и отечественной вычислительной техники после второй половины 20 века. Влияние на развитие вычислительной техники достижений, связанных с производством компьютерной элементной базы, программного обеспечения и технологий программирования.

46. В коробке 32 разноцветных карандаша. Из этой коробки достали красный карандаш. Сколько бит информации содержит это сообщение.

47. Сколько килобайт информации содержит сообщение из 384 символов 16 символьного алфавита.

48. Что такое конъюнкция, дизъюнкция. Дать определение ранга конъюнкции. Какие формулы называются ДНФ. Что такое СДНФ. Привести примеры

49. Средства разработки и отладки программ. Языки и системы программирования. Низкоуровневые и высокоуровневые языки. Алгоритмы, их обозначение и реализация.

50. Один студент, являясь резидентом разведки, решил поздравить своего друга - студента с успешной сдачей экзамена по информатике. Он подготовил текст следующей радиограммы: Дорогой Алекс! От всей души поздравляю с успешной сдачей экзамена по информатике. Желаю дальнейших успехов. Юстас. Известно, что радиопеленгатор засекает место откуда ведется передача, если передача длится не менее трех минут. Вопрос: с какой скоростью должна вестись передача данной радиограммы, чтобы ее не успели запеленговать?

51. Привести классификацию средств и методов, которые используются для защиты информации, привести примеры

52. Какие угрозы в области информационной безопасности вы знаете. Привести исторические примеры.

53. Какие правовые акты и законы существуют в области защиты информации. Кто такие пираты, хакеры, крекеры, снифферы.

54. Перечислите известные вам технические средства защиты информации, которые применяются при передаче текстовых и речевых сообщений, при защите от несанкционированного доступа. при защите информации от искажения.

25. Литература

1. Энциклопедия для детей. [Т.22] Информатика / ред. коллегия: М.Аксенова, Е.Журавлева, А.Леонов.– М.: Мир энциклопедий Аванта+, 2007. – 624 с.: ил.

2. Частиков А.П. Архитекторы компьютерного мира / А.П. Частиков . – СПб. БХВ-Петербург, 2002. – 384 с.: ил.

3. Могилев А.В. Информатика: Учеб.пособие для студ. пед. вузв / А.В.Могилев, Н.И.Пак, Е.К.Хеннер; Под ред Е.К.Хеннера. – 3-е изд., перерабо. и доп. – М.: Издательский центр "Академия", 2004. – 848 с.

4. Волкова В.Н. Искусство формализациии: От математики - к теории систем и от теории систем - к математике. Изд. 2-е. СПб.: Изд-во СПбГПУ, 2004. 199 С.

5. Советов Б.Я. Моделирование систем: Учеб. для вузов / Б.Я. Советов, С.А. Яковлев. – 5 е. изд., стер. – М.:Высш. шк., 2007 . – 373 с. ил.

6. Карпов Ю.Г. Теория автоматов / Ю.Г. Карпов – СПб.: Питер, 2002. – 224 с.: ил.

7. Бейсик. QBasic и Visual Basic / Я.Н. Глинский, В.Е. Анохин, В.А. Ряжская.- СПб.: ООО «ДиаСофтЮП», 2002. – 192 с.

8. История создания языков Pascal и Delphi [Электронный ресурс].– Режим доступа: http:// www.interestprograms.ru/ articles/historyprogramming/historypascal

9. История создания языка C++ [Электронный ресурс].– Режим доступа: https:// ru.wikipedia.org /wiki/C%2B%2B