МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 02 2018 г.

БИОЛОГИЧЕСКАЯ И МЕДИЦИНСКАЯ ИНФОРМАТИКА

Методические указания к выполнению практических занятий аспирантов направления подготовки 06.06.01. Биологические науки (Математическая биология, биоинформатика)

Курск 2018

УДК 614.2 (076.5)

Составитель Н.М. Агарков

Рецензент
доктор технических наук, профессор *Коцарь А.Г.*

Биологическая и медицинская информатика: Методические указания для выполнения практических работ аспирантов / Юго-Зап. гос. ун-т; сост.: Н.М. Агарков. - Курск, 2017. - 53с.

Содержатся методические и справочные сведения, предназначенные для практической работы аспирантов.

Методические указания по структуре, содержанию и стилю изложения материала соответствуют методическим и научным требованиям, предъявляемым к учебным и методическим пособиям.

Предназначены для аспирантов направления подготовки 06.06.01. Биологические науки.

Текст печатается в авторской редакции

Подписано в печать 14.02.18. . Формат 60x84 1/16
Усл.печ.л. 3,08. Уч.-изд.л. 2,79. Тираж 100 экз. Заказ: 1215 . Бесплатно.
Юго-Западный государственный университет.
305040. г.Курск, ул. 50 лет Октября, 94.

Практическая работа № 1 Правила двоичной арифметики и кодирование информации

Люди для записи текстовой информации используют буквы. В русском языке их 33. Комбинациями из десяти цифр (от 0 до 9) мы записываем числовые данные. При работе с графической информацией пользуемся палитрой из миллионов цветов. Наши уши различают звуки в диапазоне от 16 до 20000 Гц.

Если добавить к этому обоняние, вкусовые и тактильные ощущения, получится огромнейшее разнообразие информационных импульсов, которые может воспринимать, хранить и обрабатывать наш мозг.

При помощи технических средств невозможно воссоздать аналогичную систему работы с информацией.

Людям проще всего создавать приборы, принимающие одно из двух состояний: лампочка горит или нет, магнитное поле есть или его нет и т.д. И значительно сложнее, например, заставить лампочку в разных ситуациях светиться одним из 10 цветов. Не говоря уже о 10 миллионах цветов, воспринимаемых человеком.

В технике намного удобнее иметь дело с множеством простых элементов, чем с небольшим количеством сложных.

Чтобы иметь возможность хранить и обрабатывать информацию техническими средствами, люди решили переводить ее на максимально простой "язык", состоящий всего из двух "букв" – так называемый **двоичный** или **бинарный** код.

Идея использования бинарного кода принадлежит немецкому математику Готфриду Лейбницу (1646 - 1716).
Он разработал двоичную арифметику и даже сделал чертеж двоичной вычислительной машины, но не сумел ее построить.

Используя разные комбинации большого количества двух символов, в бинарном коде можно зашифровать любую числовую, текстовую, звуковую или графическую информацию.

Компьютер же является ничем иным, как машиной, предназначенной для хранения и обработки информации в таком виде.

Перевод данных в двоичный код называется **кодированием.**

Противоположный процесс, в результате которого бинарный код превращается в привычную для людей информацию, называется **декодированием.**

Компьютер осуществляет кодирование "на лету" при получении данных извне: ввод текста пользователем с клавиатуры, запись видео с веб-камеры, запись звука с микрофона и т.д.

Перед выводом информации на экран, в аудиосистему или же ее распечатыванием, происходит обратный процесс (декодирование).

Как осуществляется кодирование различных типов данных, рассмотрим немного ниже. Сначала давайте разберемся, из каких же символов формируется двоичный код внутри компьютера и как он там хранится.

С технической стороны компьютерный двоичный код реализуется наличием или отсутствием определенных свойств (импульсов) у мельчайших запоминающих элементов. Эти импульсы могут быть:

• *фотооптическими*

Так, поверхность любого оптического диска (CD, DVD или BluRay) состоит из спирали, которую формируют мелкие отрезки. Каждый из них может быть либо темного, либо светлого цвета. Диск быстро вращается в дисководе. На его спиральной дорожке фокусируется лазер, отражение которого попадает на фотоэлемент. Темные участки спирали поглощают свет и не передают его на фотоэлемент, светлые – наоборот, отражая свет, передают импульс фотоэлементу. В результате фотоэлемент получает информацию, зашифрованную в дорожке диска в виде темных и светлых точек.

• *магнитными*

Например, внутри жесткого диска находится быстро вращающаяся пластина. Вся ее поверхность тоже представляет собой спираль, состоящую из последовательности миллионов мелких участков. Каждый из них является элементом, который может принимать одно из двух состояний: "намагниченное", "ненамагниченное". Эти элементы и формируют двоичный код, в котором кодируется какая-то информация. Считывание состояния элементов осуществляется специальной головкой, которая быстро движется по поверхности пластины;

• *электрическими*

Например, оперативная память компьютера является микросхемой, состоящей из миллионов маленьких ячеек, созданных из микроскопических транзисторов и конденсаторов. Каждая такая ячейка может либо содержать электрический заряд, либо нет. Комбинации заряженных и разряженных ячеек оперативной памяти и формируют в ней двоичный код.

В аналогичной форме информация хранится и во всех других запоминающих микросхемах (флешки, SSD-носители и др.).

Процессор компьютера обрабатывает двоичный код тоже в виде электрических импульсов.

Иногда можно встретить ошибочное мнение, что бинарный код внутри компьютера записан в виде обычных нулей и единиц. Это следствие непонимания технической стороны вопроса. Привычных для нас *нулей и единиц в компьютере нет*. "Символами" компьютерного двоичного кода является наличие или отсутствие у мельчайшего запоминающего элемента определенного свойства (см. выше).

Чтобы было нагляднее, в учебных материалах отсутствие у элемента такого свойства *лишь условно* обозначают нулем, а его наличие – единицей. Но с таким же успехом их можно бы было обозначать точкой и тире или крестиком и ноликом.

Единицы компьютерной информации

В предыдущем пункте уже говорилось о том, что бинарный код внутри компьютера хранится в виде комбинаций большого количества элементов, каждый из которых может иметь одно из двух состояний.

Такой мельчайший элемент, участвующий в формировании бинарного кода, называется **битом.**

Битом является, например, каждая темная или светлая точка дорожки оптического диска, каждая запоминающая ячейка оперативной памяти компьютера и т.д.

Чем больше битов содержит какой-то носитель, тем больше информации на нем можно закодировать. К примеру, оптический диск типа "CD" может содержать около 6 млрд. битов. Жесткий диск - в десятки раз больше.

Но каждый отдельный бит сам по себе не имеет практической ценности. Для кодирования информации используются блоки из нескольких битов.

Представим, например, что в каком-то запоминающем устройстве содержится только один бит. В нем можно будет закодировать всего одно из двух состояний чего либо, например, одну из двух цифр или один из двух цветов. Понятное дело, что практическая ценность такого носителя минимальна.

Блок из 2 битов может принимать одно из 4 состояний:

В 3-хбитном блоке можно закодировать уже одно из 8 состояний:

Ну а 8-битный блок может принимать аж 256 разных состояний. Это уже достаточно существенная частичка двоичного кода, позволяющая отобразить один из значительного количества вариантов.

Например, каждому состоянию 8-битного блока можно сопоставить какую-то букву. Вариантов, а их 256, будет достаточно для кодирования всех русских букв, включая строчные и прописные их варианты, а также всех знаков препинания. Заменяя каждую букву соответствующим 8-мибитным блоком, из двоичного кода можно составить текст.

Этот принцип и используется для записи в компьютере текстовой информации (подробнее речь об этом пойдет ниже).

Как видите, 8-битная ячейка имеет вполне реальную практическую ценность. Поэтому ее и решили считать минимальной единицей компьютерной информации. Эта единица получила название **байт.**

Текстовые файлы состоят из сотен, тысяч или даже десятков тысяч букв. Соответственно, для их хранения в двоичном коде требуются сотни, тысячи или десятки тысяч байтов.

Поэтому на практике гораздо чаще приходится имеет дело не с байтами, а с более крупными единицами:

- *килобайтами* (1 килобайт = 1024 байт);
- *мегабайтами* (1 мегабайт = 1024 килобайт);
- *гигабайтами* (1 гигабайт = 1024 мегабайт);
- *терабайтами* (1 терабайт = 1024 гигабайт).

Кодирование числовой информации

Для работы с числовой информацией мы пользуемся системой счисления, содержащей десять цифр: 0 1 2 3 4 5 6 7 8 9. Эта система называется *десятичной*.

Кроме цифр, в десятичной системе большое значение имеют разряды. Подсчитывая количество чего-нибудь и дойдя до самой большой из доступных нам цифр (до 9), мы вводим второй разряд и дальше каждое последующее число формируем из двух цифр. Дойдя до 99, мы вынуждены вводить третий разряд. В пределах трех разрядов мы можем досчитать уже до 999 и т.д.

Таким образом, используя всего десять цифр и вводя дополнительные разряды, мы можем записывать и проводить математические операции с любыми, даже самыми большими числами.

Компьютер ведет подсчет аналогичным образом, но имеет в своем распоряжении всего две цифры - логический ноль (отсутствие у бита какого-то свойства) и логическая единица (наличие у бита этого свойства).

Система счисления, использующая только две цифры, называется *двоичной*.

При подсчете в двоичной системе добавлять каждый следующий разряд приходится гораздо чаще, чем в десятичной.

Вот таблица первых десяти чисел в каждой из этих систем счисления:

Десятичная	Двоичная	Это же число, закодированное в 1 байте запоминающего устройства
0	*0*	
1	*1*	
2	*10*	
3	*11*	
4	*100*	
5	*101*	
6	*110*	
7	*111*	
8	*1000*	
9	*1001*	

Как видите, в десятичной системе счисления для отображения любой из первых десяти цифр достаточно 1 разряда. В двоичной системе для тех же целей потребуется уже 4 разряда.

Соответственно, для кодирования этой же информации в виде двоичного кода нужен носитель емкостью как минимум 4 бита (0,5 байта).

Человеческий мозг, привыкший к десятичной системе счисления, плохо воспринимает систему двоичную. Хотя обе они построены на одинаковых

принципах и отличаются лишь количеством используемых цифр. В двоичной системе точно так же можно осуществлять любые арифметические операции с любыми числами. Главный ее минус - необходимость иметь дело с большим количеством разрядов.

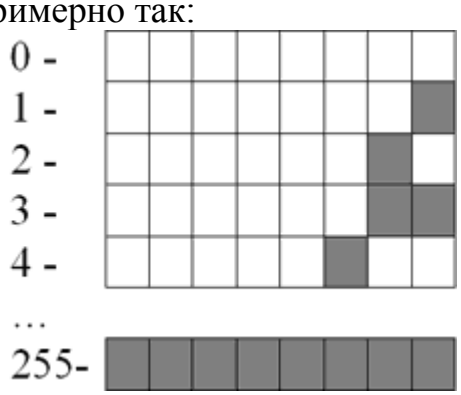
Так, самое большое десятичное число, которое можно отобразить в 8 разрядах двоичной системы - 255, в 16 разрядах – 65535, в 24 разрядах – 16777215.

Компьютер, кодируя числа в двоичный код, основывается на двоичной системе счисления. Но, в зависимости от особенностей чисел, может использовать разные алгоритмы:

• *небольшие целые числа без знака*

Для сохранения каждого такого числа на запоминающем устройстве, как правило, выделяется 1 байт (8 битов). Запись осуществляется в полной аналогии с двоичной системой счисления.

Целые десятичные числа без знака, сохраненные на носителе в двоичном коде, будут выглядеть примерно так:



• *большие целые числа и числа со знаком*

Для записи каждого такого числа на запоминающем устройстве, как правило, отводится 2-байтний блок (16 битов).

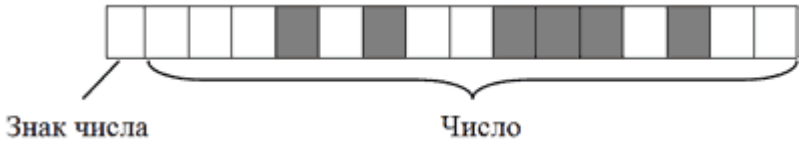
Старший бит блока (тот, что крайний слева) отводится под запись знака числа и в кодировании самого числа не участвует. Если число со знаком "плюс", этот бит остается пустым, если со знаком "минус" – в него записывается логическая единица. Число же кодируется в оставшихся 15 битах.

Например, алгоритм кодирования числа +2676 будет следующим:

1. Перевести число 2676 из десятичной системы счисления в двоичную. В итоге получится 101001110100;

2. Записать полученное двоичное число в первые 15 бит 16-битного блока (начиная с правого края). Последний, 16-й бит, должен остаться пустым, поскольку кодируемое число имеет знак +.

В итоге +2676 в двоичном коде на запоминающем устройстве будет выглядеть так:



Примечательно, что в двоичном коде присвоение числу отрицательного значения предусматривает не только изменение старшего бита. Осуществляется также инвертирование всех остальных его битов.

Чтобы было понятно, рассмотрим алгоритм кодирования числа -2676:

1. Перевести число 2676 из десятичной системы счисления в двоичную. Получим все тоже двоичное число 101001110100;

2. Записать полученное двоичное число в первые 15 бит 16-битного блока. Затем инвертировать, то есть, изменить на противоположное, значение каждого из 15 битов;

3. Записать в 16-й бит логическую единицу, поскольку кодируемое число имеет отрицательное значение.

В итоге -2676 на запоминающем устройстве в двоичном коде будет иметь следующий вид:

Запись отрицательных чисел в инвертированной форме позволяет заменить все операции вычитания, в которых они участвуют, операциями сложения. Это необходимо для нормальной работы компьютерного процессора.

Максимальным десятичным числом, которое можно закодировать в 15 битах запоминающего устройства, является 32767. Иногда для записи чисел по этому алгоритму выделяются 4-байтные блоки. В таком случае для кодирования каждого числа будет использоваться 31 бит плюс 1 бит для кодирования знака числа. Тогда максимальным десятичным числом, сохраняемым в каждую ячейку, будет 2147483647 (со знаком плюс или минус).

• *дробные числа со знаком*

Дробные числа на запоминающем устройстве в двоичном коде кодируются в виде так называемых *чисел с плавающей запятой* (точкой). Алгоритм их кодирования сложнее, чем рассмотренные выше. Тем не менее, попытаемся разобраться.

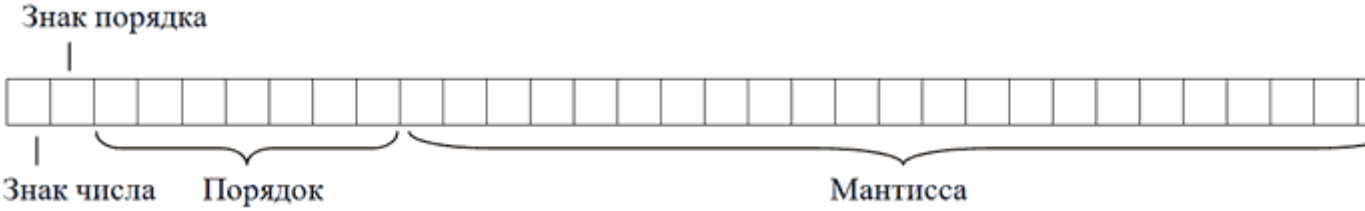
Для записи каждого числа с плавающей запятой компьютер чаще всего выделяет 4-байтную ячейку (32 бита):

• в старшем бите этой ячейки (тот, что крайний слева) записывается знак числа. Если число отрицательное, в этот бит записывается логическая единица, если оно со знаком "плюс" – бит остается пустым.

• во втором слева бите аналогичным образом записывается знак *порядка* (что такое порядок поймете позже);

• в следующих за ним 7 битах записывается значение порядка.

• в оставшихся 23 битах записывается так называемая *мантисса* числа.



Чтобы стало понятно, что такое порядок, мантисса и зачем они нужны, переведем в двоичный код десятичное число 6,25.

Порядок кодирования будет примерно следующим:

1. Перевести десятичное число в двоичное (десятичное 6,25 равно двоичному 110,01);

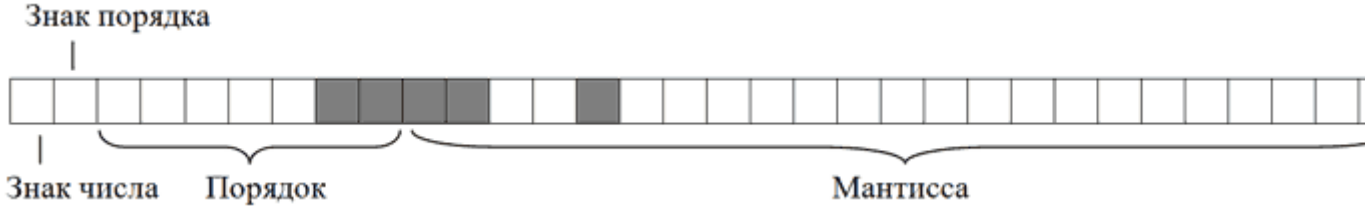
2. Определить мантиссу числа. Для этого в числе необходимо передвинуть запятую в нужном направлении, чтобы слева от нее не осталось ни одной единицы. В нашем случае запятую придется передвинуть на три знака влево. В итоге, получим мантиссу ,11001;

3. Определить значение и знак порядка.

Значение порядка – это количество символов, на которое была сдвинута запятая для получения мантиссы. В нашем случае оно равно 3 (или 11 в двоичной форме);

Знак порядка – это направление, в котором пришлось двигать запятую: влево – "плюс", вправо – "минус". В нашем примере запятая двигалась влево, поэтому знак порядка – "плюс";

Таким образом, порядок двоичного числа 110,01 будет равен +11, а его мантисса ,11001. В результате в двоичном коде на запоминающем устройстве это число будет записано следующим образом



Обратите внимание, что мантисса в двоичном коде записывается, начиная с первого после запятой знака, а сама запятая упускается.

Числа с плавающей запятой, кодируемые в 32 битах, называю *числами одинарной точности*.

Когда для записи числа 32-битной ячейки недостаточно, компьютер может использовать ячейку из 64 битов. Число с плавающей запятой, закодированное в такой ячейке, называется *числом двойной точности*.

Двоичное кодирование текстовой информации

Существует несколько общепринятых стандартов кодирования текста в двоичном коде.

Одним из наиболее "старых" (разработан еще в 1960-х гг.) является стандарт *ASCII* (от англ. American Standard Code for Information Interchange). Это 7-битный стандарт кодирования. То есть, используя его, компьютер записывает каждую букву или знак в одну 7-битную ячейку запоминающего устройства.

Как известно, ячейка из 7 битов может принимать 128 различных состояний. Соответственно, в стандарте ASCII каждому из этих 128 состояний соответствует какая-то буква, знак препинания или специальный символ.

Дальнейшее развитие компьютерной техники показало, что 7-битный стандарт кодирования является слишком "тесным". В 128 состояниях, принимаемых 7-битной ячейкой, невозможно закодировать буквы всех существующих в мире письменностей.

Поэтому разработчики программного обеспечения начали создавать собственные 8-битные стандарты кодировки текста. За счет дополнительного бита диапазон кодирования в них был расширен до 256 символов. Чтобы не было путаницы, первые 128 символов в таких кодировках, как правило, соответствуют стандарту ASCII. Оставшиеся 128 - реализуют региональные языковые особенности.

Восьмибитными кодировками, распространенными в нашей стране, являются *KOI8, UTF8, Windows-1251* и некоторые другие.

Разработаны также и универсальные стандарты кодирования текста (*Unicode*), включающие буквы большинства существующих языков. В них для записи одного символа может использоваться до 16 битов и даже больше.

Существование большого количества кодировок текста является причиной многих проблем. Вы, наверное, уже встречались с ситуацией, когда в некоторых программах на экране вместо букв отображаются непонятные

"кракозябры". Это потому, что компьютер иногда "ошибается" и неверно определяет кодировку, в которой этот текст хранится в его памяти.

В перспективе, вероятно, будет принят единый стандарт кодирования текста, полностью учитывающий разнообразие существующих письменностей, на который постепенно перейдут все компьютеры, независимо от локации и используемого программного обеспечения. Но произойдет это, судя по всему, не скоро.

Кодирование изображений в двоичный код

Чтобы сохранить в двоичном коде фотографию, ее сначала виртуально разделяю на множество мелких цветных точек, называемых *пикселями* (что-то на подобии мозаики).

После разбивки на точки цвет каждого пикселя кодируется в бинарный код и записывается на запоминающем устройстве.

Первая цифровая фотокамера, созданная в 1975 г. инженерами компании Kodak, весила 3 кг, делала черно-белые снимки размером 100X100 пикселей и сохраняла их в двоичном коде на магнитную ленту.
Запись одного снимка длилась дольше 20 секунд.

Если говорят, что размер изображения составляет, например, 512 x 512 точек, это значит, что оно представляет собой матрицу, сформированную из 262144 пикселей (количество пикселей по вертикали, умноженное на количество пикселей по горизонтали).

Прибором, "разбивающим" изображения на пиксели, является любая современная фотокамера (в том числе веб-камера, камера телефона) или сканер.

И если в характеристиках камеры значится, например, "10 Mega Pixels", значит количество пикселей, на которые эта камера разбивает изображение для записи в двоичном коде, - 10 миллионов.

Чем на большее количество пикселей разделено изображение, тем реалистичнее выглядит фотография в декодированном виде (на мониторе или после распечатывания).

Однако качество кодирования фотографий в бинарный код зависит не только от количества пикселей, но также и от их цветового разнообразия.

Алгоритмов записи цвета в двоичном коде существует несколько. Самым распространенным из них является **RGB**. Эта аббревиатура – первые буквы названий трех основных цветов: красного – англ.**R**ed, зеленого – англ. **G**reen**,** синего – англ. **B**lue**.**

Из школьных уроков рисования, Вам, наверное, известно, что смешивая эти три цвета в разных пропорциях, можно получить любой другой цвет или оттенок.

На этом и построен алгоритм RGB. Каждый пиксель записывается в двоичном коде путем указания количества красного, зеленого и синего цвета, участвующего в его формировании.

Чем больше битов выделяется для кодирования пикселя, тем больше вариантов смешивания этих трех каналов можно использовать и тем значительнее будет цветовая насыщенность изображения.

Цветовое разнообразие пикселей, из которых состоит изображение, называется **глубиной цвета.**

Если для кодирования каждого пикселя какого-то изображения выделяется 8 битов двоичного кода, цветовое разнообразие составит 256 цветов.

Глубина цвета 12-битов даст 4096 цветов, 16-битов - 65536 цветов, 18-битов - 262144 цветов.

Максимальная глубина цвета, используемая в компьютерной технике - 24 бита. Такую глубину часто называют *True Color* ("Настоящий цвет"). Она позволяет отобразить около 16,7 млн. цветов. Глаз человека не способен воспринимать большее их количество.

Тем не менее, часто встречается и так называемая 32-битная глубина цвета. Она не предусматривает увеличение количества оттенков. Дополнительные биты, выделяемые для кодирования каждого пикселя, предназначены для регулирования степени его прозрачности или же не используются.

Описанная выше техника формирования изображений из мелких точек является наиболее распространенной и называется **растровой**. Но кроме

растровой графики, в компьютерах используется еще и так называемая **векторная** графика.

Векторные изображения создаются только при помощи компьютера (фотокамеры этого делать "не умеют") и формируются не из пикселей, а из графических примитивов (линий, многоугольников, окружностей и др.).

Зачем нужна векторная графика? В известной детской песенке поется, что для изображения "человечка" достаточно нарисовать всего две "палки" и "огуречек". А представьте, насколько трудно вручную составить человечка из большого числа точек.

Векторная графика - это *чертежная графика*. Она очень удобна для компьютерного "рисования" и широко используются дизайнерами при графическом оформлении печатной продукции, в том числе создании огромных рекламных плакатов, а также в других подобных ситуациях.

Векторное изображение в двоичном коде записывается как совокупность примитивов с указанием их размеров, цвета заливки, места расположения на холсте и некоторых других свойств.

Например, чтобы записать на запоминающем устройстве векторное изображение круга, компьютеру достаточно в двоичный код закодировать тип объекта (окружность), координаты его центра на холсте, длину радиуса, толщину и цвет линии, цвет заливки.

В растровой системе пришлось бы кодировать цвет каждого пикселя. И если размер изображения большой, для его хранения понадобилось бы значительно больше места на запоминающем устройстве.

Тем не менее, векторный способ кодирования не позволяет записывать в двоичном коде реалистичные фото. Поэтому все фотокамеры работают только по принципу растровой графики. Рядовому пользователю иметь дело с векторной графикой в повседневной жизни приходится не часто.

Кодирование звуковой информации

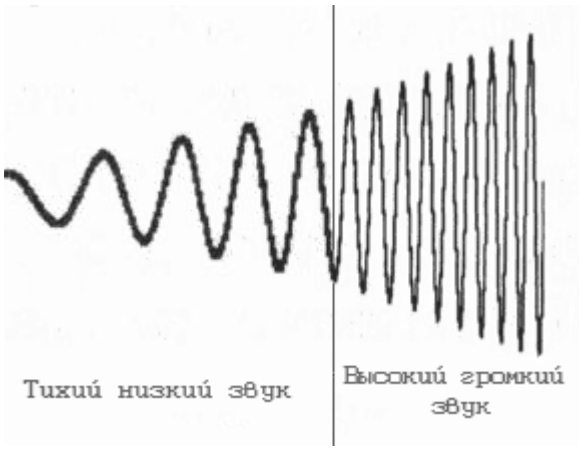
Любой звук, слышимый человеком, является колебанием воздуха, которое характеризируется двумя основными показателями: частотой и амплитудой.

Амплитуда колебаний - это степень отклонения состояния воздуха от начального при каждом колебании. Она воспринимается нами как громкость звука.

Частота колебаний - это количество отклонений состояний воздуха от начального за единицу времени. Она воспринимается как высота звука.

Так, тихий комариный писк - это звук с высокой частотой, но с небольшой амплитудой. Звук грозы наоборот имеет большую амплитуду, но низкую частоту.

Если графически изобразить звуковую волну, она будет выглядеть следующим образом:



Схему работы компьютера со звуком в общих чертах можно описать так.

Микрофон превращает колебания воздуха в аналогичные по характеристикам электрические колебания.

Контрольные вопросы

1. Правила двоичной арифметики и кодирование информации
2. двоичный или бинарный код
3. кодирование.
4. декодирование
5. Единицы компьютерной информации
6. Кодирование числовой информации
7. Двоичное кодирование текстовой информации
8. Кодирование изображений в двоичный код
9. Кодирование звуковой информации

Практическая работа № 2 Разработка и конструирование схем алгоритмов

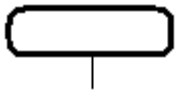
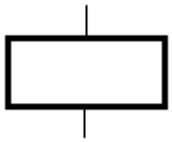
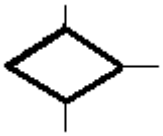
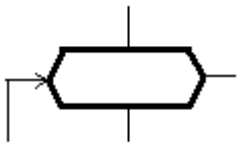
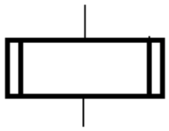
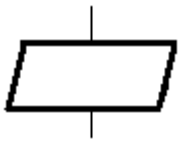
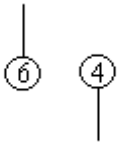
Блок-схемой будем называть такое графическое представление алгоритма, когда отдельные действия (или команды) представляются в виде геометрических фигур – *блоков*. Внутри блоков указывается информация о действиях, подлежащих выполнению. Связь между блоками изображают с помощью линий, называемых *линиями связи*, обозначающих передачу управления.

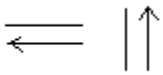
Существует Государственный стандарт, определяющий правила создания блок-схем. Конфигурация блоков, а также порядок графического оформления блок-схем регламентированы ГОСТ 19.701-90 "Схемы алгоритмов и программ". В табл. 2.1 приведены обозначения некоторых элементов, которых будет вполне достаточно для изображения алгоритмов при выполнении студенческих работ.

Правила составления блок-схем:

1. Каждая блок-схема должна иметь блок «*Начало*» и один блок «*Конец*».
2. «*Начало*» должно быть соединено с блоком «*Конец*» линиями потока по каждой из имеющихся на блок-схеме ветвей.
3. В блок-схеме не должно быть блоков, кроме блока «*Конец*», из которых не выходит линия потока, равно как и блоков, из которых управление передается «в никуда».
4. Блоки должны быть пронумерованы. *Нумерация* блоков осуществляется сверху вниз и слева направо, номер блока ставится вверху слева, в разрыве его начертания.
5. Блоки связываются между собой линиями потока, определяющими последовательность выполнения блоков. Линии потоков должны идти параллельно границам листа. *Если линии идут* ***справа налево*** *или* ***снизу вверх****, то стрелки в конце линии обязательны*, в противном случае их можно не ставить.
6. По отношению к блокам линии могут быть *входящими* и *выходящими*. Одна и та же линия потока является выходящей для одного блока и входящей для другого.
7. От блока «*Начало*» в отличие от всех остальных блоков линия потока только выходит, так как этот блок – первый в блок-схеме.
8. Блок «*Конец*» имеет только вход, так как это последний блок в блок-схеме.
9. Для простоты чтения желательно, чтобы линия потока входила в блок «Процесс» сверху, а выходила снизу.
10. Чтобы не загромождать блок-схему сложными пересекающимися линиями, линии потока можно разрывать. При этом в месте разрыва ставятся *соединители*, внутри которых указываются номера соединяемых блоков. В блок-схеме не должно быть разрывов, не помеченных соединителями.

11.Чтобы не загромождать блок, можно информацию о данных, об обозначениях переменных и т.п. размещать в *комментариях* к блоку.

Название блока	**Обозначение блока**	**Назначение блока**
1	**2**	**3**
Терминатор		Начало/Конец программы или подпрограммы
Процесс		Обработка данных (вычислительное действие или последовательность вычислительных действий)
Решение		Ветвление, выбор, проверка условия. В блоке указывается условие или вопрос, который определяет дальнейшее направление выполнения алгоритма
Подготовка		Заголовок счетного цикла
Предопределенный процесс		Обращение к процедуре
Данные		Ввод/Вывод данных
Соединитель		Маркировка разрыва линии потока
Комментарий		Используется для размещения пояснений к действиям

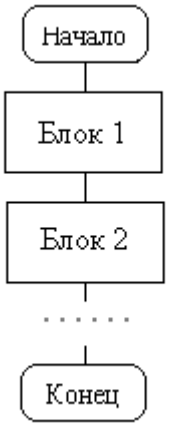
Горизонтальные и вертикальные потоки		Линии связей между блоками, направление потоков

Типы алгоритмов

Тип алгоритма определяется характером решаемой в соответствии с его командами задачи. Различают три типа алгоритмов: линейные, разветвляющиеся, циклические.

Линейный алгоритм состоит из упорядоченной последовательности действий, не зависящей от значений исходных данных, при этом каждая команда выполняется только один раз строго после той команды, которая ей предшествует.

Таким, например, является алгоритм вычисления по простейшим безальтернативным формулам, не имеющий ограничений на значения входящих в эти формулы переменных. Как правило, линейные процессы являются составной частью более сложного алгоритма.



Разветвляющимися называются алгоритмы, в которых в зависимости от значения какого-то выражения или от выполнения некоторого логического условия дальнейшие действия могут производиться по одному из нескольких направлений.

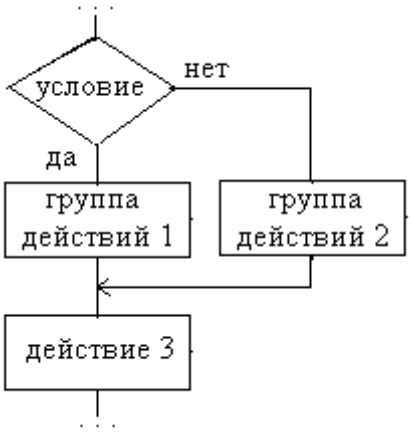
Каждое из возможных направлений дальнейших действий называется *ветвью*.

В блок-схемах разветвление реализуется специальным блоком «*Решение»*. Этот блок предусматривает возможность двух выходов. В самом блоке «Решение» записывается логическое условие, от выполнения которого зависят дальнейшие действия.

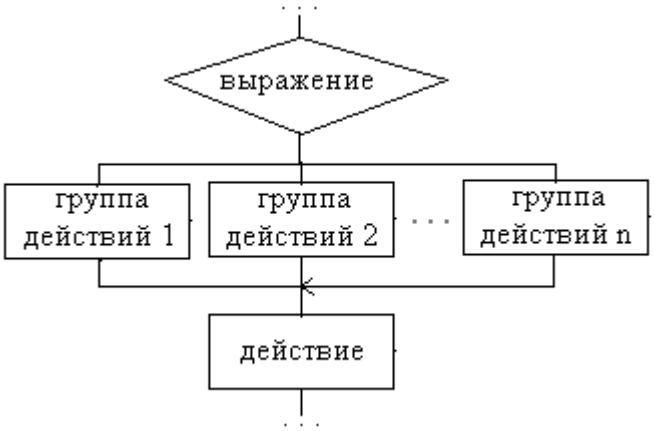
Различают несколько видов разветвляющихся алгоритмов.

1. ***«Обход»*** – такое разветвление, когда одна из ветвей не содержит ни одного оператора, т.е. как бы обходит несколько действий другой ветви.

2. ***«Разветвление»*** – такой тип разветвления, когда в каждой из ветвей содержится некоторый набор действий.



3. ***«Множественный выбор»*** – особый тип разветвления, когда каждая из нескольких ветвей содержит некоторый набор действий. Выбор направления зависит от значения некоторого выражения.



Циклические алгоритмы применяются в тех случаях, когда требуется реализовать многократно повторяющиеся однотипные вычисления. *Цикл* – это последовательность действий, которая может выполняться многократно, т.е. более одного раза.

Различают:

- циклы с известным числом повторений (или со счетчиком);
- циклы с неизвестным числом повторений (циклы с предусловием и циклы с постусловием).

В любом цикле должна быть переменная, которая управляет выходом из цикла, т.е. определяет число повторений цикла.
Последовательность действий, которая должна выполняться на каждом *шаге цикла* (т.е. при каждом повторении цикла), называется *телом цикла* или *рабочей частью цикла*.

Контрольные вопросы

1. Блок-схема
2. Правила составления блок-схем
3. Типы алгоритмов
4. Циклы
5. Направление движения

Практическая работа № 3 Программирование разветвляющихся и простых циклических вычислительных и итерационных процессов

Программирование разветвляющихся и циклических вычислительных процессов.

Вычислительные процессы, в которых часть вычислений повторяется многократно, называются *циклическими процессами*.

Виды операторов цикла.

1. Оператор цикла с предварительным условием.
2. Оператор цикла с последующим условием.
3. Оператор цикла с параметром.

Оператор цикла с предварительным условием используется в тех случаях, когда заранее неизвестно число повторений цикла.

Форма записи оператора цикла с предусловием:

while логическое выражение *do*

begin операторы циклической части программы (тело цикла) *end*;

Если в циклической части оператора стоит всего один оператор, то операторные скобки *begin* и *end* можно не указывать.

Оператор цикла с предварительным условием действует следующим образом. Предварительно проверяется логическое выражение. Если оно истинно, то выполняются операторы циклической части программы. Если логическое выражение ложно, то происходит выход из цикла.

Если с самого начала значение логического выражения не является истиной, то операторы циклической части не выполняются ни разу.

Логическое выражение должно меняться в теле цикла, иначе цикл будет бесконечным и через некоторое время компьютер зависнет.

Пример. Вычислить значения функции Y при изменении X от C до D с шагом H.

$$Y = \begin{cases} X \cdot COS(X), если X > A; \\ SIN(X) + A \cdot X, если X < A \end{cases}$$

Программа

```
var
X, A. Y, C, D, H : real;
begin
writeln (‘Введите C, D, H, A’);
readln (C, D, H, A);
x:= C;
while x <= D do
begin
if x > A then y:= x*cos(A*x)
else y:=sin(x) + A*a;
writeln (‘y= ‘,y:10:2, ‘ x= ‘,x:10:2);
x:= x + H
```

end;
end.
Оператор цикла с последующим условием имеет следующую форму записи:
repeat
тело цикла (выполняемые операторы)
until логическое выражение.
Данный оператор организует выполнение цикла, состоящего из любого числа операторов, с неизвестным заранее числом повторений. Тело цикла выполняется хотя бы один раз. Выход из цикла осуществляется при истинном значении некоторого логического выражения. Истинность логического выражения проверяется в конце каждой итерации.
Пример. Вычислить значения функции Y при изменении X от C до D с шагом H.

$$Y = \begin{cases} X \cdot COS(X), если X > A; \\ SIN(X) + A \cdot X, если X < A \end{cases}$$

Программа
var
X, A. Y, C, D, H : *real*;
begin
writeln (‘Введите C, D, H, A’);
readln (C,D,H,A);
x:= C;
repeat
if x > A *then* y:= x*cos(A*x)
else y:= sin(x) + A*a;
writeln (‘y= ‘,y:10:2, ‘ x= ‘,x:10:2);
x:=x + H
until x<=D
end.
Оператор цикла с параметром организует выполнение одного оператора заранее известное число раз.
Существует два варианта оператора цикла с параметром.

1. *for* i:= a1 *to* a2 *do*

begin
тело цикла
end;

2. *fo*r i:= a1 *downto* a2 *do*

begin
тело цикла
end,

где: i - параметр цикла, являющийся переменной порядкового типа; a1 - выражение, определяющее начальное значение параметра цикла; a2 - выражение, определяющее конечное значение параметра цикла.

Цикл действует следующим образом. Вначале вычисляются и запоминаются начальное и конечное значения параметра цикла. Далее параметру i цикла присваивается начальное значение. Затем значение параметра i цикла сравнивается со значением a2. Пока параметр цикла остается меньше или равен конечному значению a2 (в первом варианте) и больше или равен конечному значению a2 (во втором варианте) выполняется в очередной раз операторы из тела цикла. В противном случае происходит выход из цикла. После выхода из цикла параметр цикла становится неопределенным.

Пример. Вычислить значения функции Y при изменении X от C до D с шагом 1.

$$Y = \begin{cases} X \cdot COS(X), если X > A; \\ SIN(X) + A \cdot X, если X < A \end{cases}$$

Программа.

```
var
X, A. Y, C, D : real;
begin
writeln (‘Введите C, D, H, A’);
readln (C,D,H,A);
for i:=C to D do begin if x > A then y:= x*cos(A*x)
else y:=sin(x) + A*a;
writeln (‘y= ‘,y:10:2, ‘ x= ‘,x:10:2);
x:=x + H;
end;
readln
end.
```

Контрольные вопросы

1. Программирование разветвляющихся процессов
2. Программирование простых циклических вычислительных процессов
3. Программирование итерационных процессов
4. Оператор цикла с предварительным условием

Практическая работа № 4 Программирование простых циклических вычислительных процессов в одномерном массиве и сложных циклических вычислительных процессов в матрице

Краткие теоретические сведения

Операторы циклов применяют, когда надо повторить некоторые действия (операторы и операции) несколько раз, и такие участки алгоритмов называют циклы.

Оператор цикла for

Основная форма оператора цикла for имеет вид:

for (выражение_1; выражение_2; выражение_3)
оператор;

где **выражение_1** – инициализация начального значения параметра цикла;
выражение_2 – проверка условия на продолжение цикла;
выражение_3 – изменение параметра цикла (коррекция параметра);
оператор – простой или составной оператор языка С.

Схема работы оператора следующая: только один раз вначале вычисляется выражение_1, затем проверяется выражение_2, и если оно - «истина», то выполняется циклический участок программы, затем производится коррекция параметра, и так до тех пор, пока выражение_2 не примет значение «ложь».

Например: for (k=1; k<5; k++)
printf("\n %d", k);

В результате выполнения этого оператора печатаются в столбик цифры от 1 до 4.

В качестве параметра цикла можно использовать переменную любого базового типа.

Например: for(ch='a'; ch<='z'; ch++) /* вывод на экран БУКВ */
printf(" %c",ch); /* латинского алфавита */

Необходимо тщательно контролировать структуру циклов for в программе, чтобы не получился бесконечный цикл (из которого нет выхода).

Например: for(k=10; k>6;k++)
printf("бесконечный цикл\n");

Выйти из цикла досрочно можно следующими способами:
- по дополнительному условию;
- используя операторы:
  - **break;** - завершения работы цикла, в котором находится break, управление передается на первый после цикла выполняемый оператор;
  - **exit(int Kod);** - происходит выход из программы;
  - **return;** - осуществляется выход из функции;
  - с помощью оператора безусловного перехода **goto <метка>;**

Досрочное **завершение текущего циклического шага** возможно при помощи дополнительного условия или оператора **continue**, который прерывает выполнение текущего шага цикла, т.е. пропускает операторы

оставшейся части цикла и передает управление в головной оператор цикла для коррекции параметра и проверки условия.
Передавать управление извне вовнутрь цикла запрещается.
Любое из выражений цикла for в круглых скобках может отсутствовать, но символ «;» опускать нельзя.
Например: int i=0;
for(; i<3; i++)
puts(“Hello!”);
Циклические операторы while и do–while
Основная форма циклического оператора **while**:
while (условие)
оператор;
где **оператор** – это простой, составной или пустой оператор.
Цикл выполняется до тех пор, пока условие принимает значение «истина», т.е. выражение в скобках возвращает ненулевой результат. Это цикл с предусловием – сначала проверяется условие, затем выполняется оператор. Поэтому цикл while не выполнится ни разу, если изначально результат вычисления условия будет равен 0.
Основная форма оператора **do – while**:
do
оператор;
while (условие);
где **оператор** – это простой, составной или пустой оператор.
Оператор do–while – оператор цикла с постусловием, т.е. сначала выполняется оператор, а затем проверяется условие на истинность. Так как в цикле do–while условие проверяется в конце цикла, то цикл будет выполнен хотя бы один раз.
В циклах типа while и do–while допустимы те же способы досрочного выхода из цикла и досрочное завершение текущего шага цикла, как и в операторе for, но в последнем случае в отличие от цикла for управление передается на проверку условия. Для предотвращения бесконечного цикла, внутри циклов while и do–while нужно предусмотреть изменение переменных, входящих в условие.
Например:
```
int i;
for (i=1;i<=300;i++) /* печать целых чисел, кратных 5 */
{
if (i%5!=0) continue;
printf(“%5d”,i);
}
```
Контрольные вопросы

1. Программирование простых циклических вычислительных процессов в одномерном массиве и
2. Программирование сложных циклических вычислительных процессов в матрице

Практическая работа № 5 Обработка строковых данных и программирование сложных циклических вычислительных процессов табулирования функций и вычисления сумм

Итак, основное назначение циклов -- обработка большого объема данных. Математически эта обработка зачастую сводится к поиску, выбору и статистической обработке нужных величин. Практически в любой реальной задаче мы ищем максимальные и минимальные значения в наборе данных, суммируем или перемножаем требуемые данные, определяем арифметическое среднее или количество элементов, отвечающих условию. Для решения всех этих распространенных задач существуют *типовые алгоритмы*, задающие правила выполнения соответствующих расчетов. Изучением этих алгоритмов мы займемся в гл. 11 и 12.

Алгоритм табулирования

Применяется для составления всевозможных таблиц, которыми могут быть как абстрактная таблица значений математической функции, так и конкретная таблица стоимости товара или платежей, совершенных абонентом сотового оператора.

В общем виде алгоритм можно описать так:

1. до цикла задается начальное значение управляющей переменной, условием выхода из цикла служит достижение управляющей переменной конечного значения;

2. в теле цикла на каждом шаге вычисляется очередное значение функции, зависящее от управляющей переменной, затем формируется строка таблицы;

3. в конце шага цикла значение управляющей переменной (обозначим ее x) изменяется оператором вида x:=x+d;, где d -- заданный шаг по управляющей переменной.

В качестве примера составим таблицу синусов в пределах от 0 до π с шагом по аргументу 0.25. Обозначим аргумент как x, значение синуса от x обозначим как y. В простейшем случае программа табулирования может выглядеть так:

```
var x,y:real;
begin
 writeln('x':10,'sin(x)':10);
  {печать заголовка таблицы до цикла}
 x:=0; {начальное значение аргумента}
 while x<=pi+1e-6 do begin
   y:=sin(x); {вычисление функции}
   writeln (x:10:2, y:10:2);
  {печать строки таблицы}
   x:=x+0.25; {шаг по x}
 end;
end.
```

"Расширим" задачу за счет использования произвольных границ изменения аргумента и произвольного шага, а также выполнения всех необходимых

проверок корректности. Пусть, например, требуется составить таблицу значений следующей функции:

$$f(x)=\begin{cases} x^3, & \text{если} \quad x \le 0 \\ \sqrt[3]{|x|}, & \text{если} \quad x > 0 \end{cases}, \quad x \in [a,b], \quad \Delta x = 0.1,$$ значения a, b вводятся пользователем.

Напишем текст программы, сопроводив его соответствующими комментариями.

```
var x,f,a,b,dx:real;
 n:integer; {счетчик выведенных строк}
begin
 repeat {Цикл ввода с контролем
  правильности значений: a,dx,b должны быть
  числами, dx>0, a+dx должно быть меньше b}
  writeln ('Введите a,dx,b:');
  {$I-}read (a,dx,b);{$I+}
  if IoResult <> 0 then begin
   writeln ('Вы не ввели 3 числовых ',
  'значения, попробуем еще раз');
   continue;
  end;
  if (dx<=0) or (a+dx>=b) then begin
   writeln ('Вы не ввели допустимые ',
    'данные, попробуем еще раз');
   continue;
  end
  else break;
 until false;
 {Печать заголовка таблицы}
 writeln;
 writeln ('x':10,'f(x)':10);
 x:=a;
 n:=2; {2 строки уже использованы}
 while x<=b+1e-6 do begin
   {в условии цикла учитываем возможную
    погрешность работы с real!}
  if x<=0 then f:=sqr(x)*x
  else f:=exp(1/3*ln(abs(x)));
   {корень 3 степени взяли через exp и ln}
  writeln (x:10:2,f:10:2);
  n:=n+1;
  if n=24 then begin
  {На экране консоли по умолчанию 25 строк}
   write ('Нажмите Enter...');
```

```
   reset (input); readln;
   n:=1;
  end;
  x:=x+dx;
 end;
 writeln ('Таблица выведена');
 reset (input); readln;
end.
```

Как видно из примера, основной порядок действий -- такой же, как в предыдущей задаче. Так как экран консоли по умолчанию содержит всего 25 строк, с помощью переменной n мы дополнительно контролируем число уже выведенных строк и делаем по заполнении экрана паузу до нажатия пользователем клавиши Enter.

Разумеется, другие изученные нами виды циклов также могут применяться при табулировании. Рассмотрим в качестве примера следующую задачу.

Известна стоимость единицы товара. Составить таблицу стоимости 1, 2, ..., K единиц товара, значение K вводится.

Так как число единиц товара -- заведомо целое, при программировании задачи будет удобен цикл for:

```
 var t:real;
    i,k:integer;
begin
 writeln;
 writeln ('Стоимость единицы товара:');
 read (t);
 writeln ('Количество единиц товара:');
 read (k);
 writeln ('Единиц':10,'Стоимость':10);
 for i:=1 to k do
  writeln (i:10,(i*t):10:2);
end.
```

Здесь для простоты мы исключили сделанные в предыдущем примере проверки. Стоимость единицы товара обозначена t, переменная i необходима для перебора возможных значений единиц товара в цикле for. Поскольку счетчик цикла for автоматически меняется с шагом 1, а оператором writeln можно выводить не только значения переменных, но и выражения, основной цикл программы состоит из одного оператора и не нуждается в операторных скобках.

Контрольные вопросы

1. Обработка строковых данных и
2. Программирование сложных циклических вычислительных процессов
3. Табулирование функций и вычисления сумм
4. Алгоритм табулирования

Практическая работа № 6 Программирование файловых операций. Графические операторы, элементарные построения. Построение динамически изменяющихся объектов

Статическими величинами называются такие, память под которые выделяется во время компиляции и сохраняется в течение всей работы программы.

В языках программирования (Pascal, C, др.) существует и другой способ выделения памяти под данные, который называется динамическим. В этом случае память под величины отводится во время выполнения программы. Такие величины будем называть динамическими. Раздел оперативной памяти, распределяемый статически, называется статической памятью; динамически распределяемый раздел памяти называется динамической памятью (динамически распределяемой памятью).

Использование динамических величин предоставляет программисту ряд дополнительных возможностей. Во-первых, подключение динамической памяти позволяет увеличить объем обрабатываемых данных. Во-вторых, если потребность в каких-то данных отпала до окончания программы, то занятую ими память можно освободить для другой информации. В-третьих, использование динамической памяти позволяет создавать структуры данных переменного размера.

Работа с динамическими величинами связана с использованием еще одного типа данных — ссылочного типа. Величины, имеющие ссылочный тип, называют указателями.

Указатель содержит адрес поля в динамической памяти, хранящего величину определенного типа. Сам указатель располагается в статической памяти.

Адрес величины — это номер первого байта поля памяти, в котором располагается величина. Размер поля однозначно определяется типом.

Величина ссылочного типа (указатель) описывается в разделе описания переменных следующим образом (Pascal):

Var <идентификатор> : ^<имя типа>;

Вот примеры описания указателей:

```
Type Mas1 = Array[1..100] Of Integer;
Var P1 : ^Integer;
P2 : ^String;
Pm : ^Mas1;
```

Здесь P1 — указатель на динамическую величину целого типа; P2 — указатель на динамическую величину строкового типа; Pm — указатель на динамический массив, тип которого задан в разделе Type.

Сами динамические величины не требуют описания в программе, поскольку во время компиляции память под них не выделяется. Во время компиляции память выделяется только под статические величины. Указатели — это статические величины, поэтому они требуют описания.

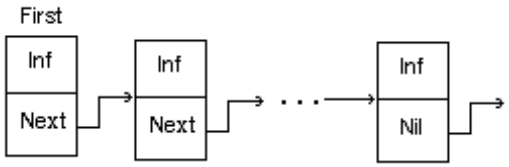
Память под динамическую величину, связанную с указателем, выделяется в результате выполнения стандартной процедуры NEW. Формат обращения к этой процедуре:

NEW(<указатель>);
Считается, что после выполнения этого оператора создана динамическая величина, имя которой имеет следующий вид:
<имя динамической величины> := <указатель>^
Пусть в программе, в которой имеется приведенное выше описание, присутствуют следующие операторы:
NEW(P1); NEW(P2); NEW(Pm);
После их выполнения в динамической памяти оказывается выделенным место под три величины (две скалярные и один массив), которые имеют идентификаторы:
P1^, P2^, Pm^
Дальнейшая работа с динамическими переменными происходит точно так же, как со статическими переменными соответствующих типов. Им можно присваивать значения, их можно использовать в качестве операндов в выражениях, параметров подпрограмм и пр. Например, если переменной P1^ нужно присвоить число 25, переменной P2^ присвоить значение символа "Write", а массив Pm^ заполнить по порядку целыми числами от 1 до 100, то это делается так:
P1^ := 25;
P2^ := 'Write';
For I := 1 To 100 Do Pm^[I] := I;
Кроме процедуры NEW значение указателя может определяться оператором присваивания:
<указатель> := <ссылочное выражение>;
В качестве ссылочного выражения можно использовать

- указатель;
- ссылочную функцию (т.е. функцию, значением которой является указатель);
- константу Nil.

Nil — это зарезервированная константа, обозначающая пустую ссылку, т.е. ссылку, которая ни на что не указывает. При присваивании базовые типы указателя и ссылочного выражения должны быть одинаковы. Константу Nil можно присваивать указателю с любым базовым типом.
До присваивания значения ссылочной переменной (с помощью оператора присваивания или процедуры NEW) она является неопределенной.
Ввод и вывод указателей не допускается.
Рассмотрим пример. Пусть в программе описаны следующие указатели:
Var D, P : ^Integer;
K : ^Boolean;
Тогда допустимыми являются операторы присваивания
D := P; K := Nil;
поскольку соблюдается принцип соответствия типов. Оператор K := D ошибочен, т.к. базовые типы у правой и левой части разные.

Если динамическая величина теряет свой указатель, то она становится "мусором". В программировании под этим словом понимают информацию, которая занимает память, но уже не нужна.
В Паскале имеется стандартная процедура, позволяющая освобождать память от данных, потребность в которых отпала. Ее формат:
DISPOSE(<указатель>);
Например, если динамическая переменная P^ больше не нужна, то оператор
DISPOSE(P)
удалит ее из памяти. После этого значение указателя P становится неопределенным. Особенно существенным становится эффект экономии памяти при удалении больших массивов.
В версиях Турбо-Паскаля, работающих под операционной системой MS DOS, под данные одной программы выделяется 64 килобайта памяти (или, если быть точнее, 65520 байт). Это и есть статическая область памяти. При необходимости работать с большими массивами информации этого может оказаться мало. Размер динамической памяти — много больше (сотни килобайт). Поэтому использование динамической памяти позволяет существенно увеличить объем обрабатываемой информации.
Следует отчетливо понимать, что работа с динамическими данными замедляет выполнение программы, поскольку доступ к величине происходит в два шага: сначала ищется указатель, затем по нему — величина.
К динамическим структурам относят:
Списки -



Стеки - структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого *вершиной стека*. По определению, элементы извлекаются из стека в порядке, обратном их добавлению в эту структуру, т.е. действует принцип "последний пришёл — первый ушёл".
Очередь — это информационная структура, в которой для добавления элементов доступен только один конец, называемый *хвостом*, а для удаления — другой, называемый *головой*. В англоязычной литературе для обозначения очередей довольно часто используется аббревиатура FIFO (first-in-first-out — первый вошёл — первым вышел).
Дерево — это совокупность элементов, называемых *узлами* (при этом один из них определен как *корень*), и отношений (родительский–дочерний), образующих иерархическую структуру узлов. Узлы могут являться величинами любого простого или структурированного типа, за исключением файлового. Узлы, которые не имеют ни одного последующего узла, называются *листьями*. В *линейном списке* каждый элемент связан со

следующим и, возможно, с предыдущим (односвязный и двусвязный). Если последний элемент связать указателем с первым, получится кольцевой список.
Каждый элемент списка содержит ключ, идентифицирующий этот элемент. Ключ обычно бывает либо целым числом, либо строкой и является частью поля данных.
Стек является простейшей динамической структурой. LIFO. Память под локальные переменные выделяется по принципу LIFO. Стеки широко применяются в СПО, компиляторах, в различных рекурсивных алгоритмах.
Очередь — это динамическая структура данных, добавление элементов в которую выполняется в один конец, а выборка — из другого конца. FIFO. Применяются при буферизованном вводе-выводе или диспетчеризации задач в операционной системе.

3. Объектно-ориентированное программирование. Объекты, свойства, методы, события. Программирование по событиям. Наследование и инкапсуляция.

Объект — элементарная единица в объектно-ориентированном программировании, заключающая в себе как описывающие объект данные, так и средства обработки этих данных.
Объектно-ориентированное программирование (ООП) — это
1)метод программирования, при использовании которого главными элементами программ являются объекты. Примерами объектно-ориентированных сред могут служить Microsoft Visual C++, Borland Delphi, Borland C++ Builder
2) методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.
Таким образом, в ООП центральное место занимают объекты, которые объединяют в одно целое (инкапсулируют) свойства объекта и возможные над ним операции (методы).
Если говорить образно, то объекты - это существительные.
Объектом являются, например, графический примитив Окружность.
Свойства объекта, т.е. его качества и характеристики (например, координаты, цвет, радиус) - это прилагательные.
Методы объекта, т.е. набор операций, которой он может выполнять (например, переместить, изменить цвет) - это глаголы.
Классы – это объединение объектов, инкапсулирующих одинаковый перечень свойств операций.
Экземпляр класса – это каждый объект по отдельности. Экземпляры могут иметь различные значения свойств, но все экземпляры одного класса имеют одинаковые методы..
Такой подход объективно обусловлен тем, что окружающий нас мир состоит из целостных объектов, которые обладают определенными свойствами и поведением. В технологии объектно-ориентированного программирования

объекты сохраняют свою целостность, все свойства объекта и его поведение описываются внутри самого объекта.

В основе объектно-ориентированного подхода лежат три понятия:

- инкапсуляция: объединение данных с процедурами и функциями в рамках единого целого — объекта;
- наследование: возможность построения иерархии объектов с использованием наследования их характеристик;
- полиморфизм: задание одного имени действию, которое передается вверх и вниз по иерархии объектов, с реализацией этого действия способом, соответствующим каждому объекту в иерархии.

Инкапсуляция. В ООП объект представляет собой запись, которая служит «оболочкой» для соединения связанных между собой данных и процедур. Другими словами, объект обладает определенными свойствами и поведением. Рассмотрим в качестве примера кнопку — типичный объект, присутствующий в интерфейсе большого количества программ. Кнопка обладает определенным поведением: она может быть нажата, после нажатия на кнопку будут происходить определенные события и т. д. Соединение таких свойств и поведения в одном объекте и называется инкапсуляцией. При описании структуры класса используются спецификаторы доступа (в C++: private, public и protected) которые определяют область видимости элементов класса. Элементы, описанные после спецификатора PRIVATE, видимы только внутри класса. Таким образом, ООП даёт возможность при описании класса задать его открытый интерфейс и скрыть его реализацию. С помощью ограничения областей видимости можно защитить программу от неверного действия пользователей или других программ. Так же может быть использован спецификатора доступа PROTECTED (он используется при наследовании, доступ к protected-членам родительского класса имеют только его потомки)

```
class A
{ public:
int a, b; //данные открытого интерфейса
int Method1 (); //метод открытого интерфейса
private:
intAa,Ab; //скрытые данные
voidMethod2 (); //скрытый метод };
```

Наследование. Объекты могут наследовать свойства и поведение от других объектов, которые называются «родительскими объектами». Это понятие можно хорошо проиллюстрировать опять на примере интерфейса программы. Возьмем в качестве «родительского объекта» самое простое окно, при вызове которого запускается определенная процедура. Классическим примером иерархии элементов GUI является окно->элементы интерфейса (кнопка, список, текстовое поле).

Окно обладает стандартными членами-функциями для своего отображения, которые перегружаются (overload) в потомках.

```
class A{ //базовый класс };
```

classB:publicA{ //publicнаследование }
class C : protected A{ //protected наследование }
class Z : private A{ //private наследование }
Полиморфизм — это слово из греческого языка, означающее «много форм». Перечень интерфейсных кнопок различных типов (простая кнопка, радиокнопка, кнопка-переключатель и т. д.) представляет собой хороший пример полиморфизма. Каждый тип объекта в этом перечне представляет собой различный тип интерфейсной кнопки. Можно описать метод для каждой кнопки, который изобразит этот объект на экране. В терминах объектно-ориентированного программирования можно сказать, что все эти типы кнопок имеют способность изображения самих себя на экране. Обычно под полиморфизмом понимают ДИНАМИЧЕСКОЕ (run-time) определение, того какой именно метод должен использоваться, т.е. во время выполнения определяется к какому конкретно типу относится объект для которого будет вызываться метод с указанным именем. Обычно это реализуется с помощью доступа к объектам дочерних классов через указатель на базовый класс (механизм виртуальных функций)
Однако способ (процедура), которым каждая кнопка должна изображать себя на экране, является различным для каждого типа кнопки. Простая кнопка рисуется на экране с помощью процедуры «вывод изображения простой кнопки», радиокнопка рисуется на экране с помощью процедуры «вывод изображения радиокнопки» и т. д.
Таким образом, существует единственное для всего перечня интерфейсных кнопок действие (вывод изображения кнопки на экран), которое реализуется специфическим для каждой кнопки способом. Это и является проявлением полиморфизма.
Объекты взаимодействуют между собой, посылая и получая сообщения.
Сообщение – это запрос на выполнение действий, содержащий набор необходимых параметров. Механизм сообщений реализуется с помощью вызова соответствующих функций. Таким образом, с помощью ООП реализуется так называемая событийно-управляемая модель.
Примером реализации может служить любая программа, управляемая с помощью меню. После запуска такая программа пассивно ожидает действия пользователей и должна уметь правильно отреагировать на любое из них. Программирование по событиям строится как противоположность традиционной (директивной) модели управления: программа после старта сама предлагает пользователю выполнить действия (ввести данные, выбрать режим) в соответствии с жёстко-заданным алгоритмом. Наступление события (например, нажатие кнопки) вызывает код обработки этого события (например, Кнопка1.ButtonClick()). Предполагается, что событие может произойти в произвольном порядке в случайные моменты времени, и программа должна уметь правильно их обрабатывать.
События и обработчики событий.
Вызов метода, взаимодействие объектов м.б. рассмотрены как отправка сообщения объекту.

Сообщение – это запрос на выполнение действий, содержащий набор необходимых параметров. Механизм сообщений реализуется с помощью вызова соответствующих функций.
Событие – это действие или инцидент обнаруженный программой.
Большинство современных приложений проектируются так, чтобы реагировать на события.
Таким образом, с помощью ООП реализуется так называемая *событийно-управляемая модель.*
Событие – это указатель на метод определенного экземпляра класса.
Отвечающий код – это обработчик событий, пишется разработчиком приложения.
Пример: программа, управляемая с помощью меню. После запуска программа пассивно ожидает действий пользователя и должна уметь правильно отреагировать на любое из них.
Программирование по событиям строится как противоположность традиционной (директивной) модели управления: программа после старта сама предлагает пользователю выполнить действия (ввести данные, выбрать режим) в соответствии с жёстко-заданным алгоритмом.
Наступление события (например, нажатие кнопки) вызывает код обработки этого события (например, Кнопка1.ButtonClick()). Предполагается, что событие может произойти в произвольном порядке в случайные моменты времени, и программа должна уметь правильно их обрабатывать.

Контрольные вопросы

1. Программирование файловых операций.
2. Графические операторы, элементарные построения.
3. Построение динамически изменяющихся объектов

Практическая работа № 7 Автоматизированная консультативная система диагностики бронхолегочных заболеваний

1. Бронхиты

Классификация бронхитов (1981)

Острый (простой) бронхит

Острый обструктивный бронхит

Острый бронхиолит

Рецидивирующий бронхит, обструктивный и необструктивный

По течению:

обострение,

ремиссия

1.1. Острый (простой) бронхит - это, как правило, проявление респираторной вирусной инфекции. Общее состояние больных нарушено незначительно. Характерны кашель, повышение температуры в течение 2-3 дней, может быть более 3 дней (длительность температурной реакции определяется основным вирусным заболеванием). Перкуторные изменения в легких отсутствуют.

Аускультативно-распространенные (рассеянные) сухие, крупно- и среднепузырчатые влажные хрипы. Длительность заболевания 2-3 недели.

Методы обследования: больные острым бронхитом в рентгенологическом и лабораторном обследовании в большинстве случаев не нуждаются. Рентгенография грудной клетки и анализ крови необходимы при подозрении на пневмонию.

Лечение больных бронхитом проводится на дому. Госпитализации требуют дети раннего возраста и больные со стойкой температурной реакцией. Дети находятся в постели 1-2 дня, при невысокой температуре можно разрешить общий режим. Лечебный стол 15 иди 16 (в зависимости от возраста). Питьевой режим с достаточным введением жидкости; компоты, морсы, вода, сладкий чай, оралит, детям старшего возраста - теплое молоко с боржомом.

Медикаментозная терапия направлена на уменьшение и облегчение кашля. С целью уменьшения кашля назначаются:

- либексин 26-60 мг в сутки, т.е. 1/4-1/2 таблетки 3-4 раза в день проглатывать не разжевывая);
- тусупрекс 6-10 мг в сутки, т.о. 1/4-1/2 таблетки 3-4 раза в день или сироп тусупрекса 1/2-1 ч.л. (в 1 ч.л. - 6 мл);
- глаувент 10-25 мг, т.е. 1/1--1/2 таблетки 2-3 раза в день после еды.

Облегчают кашель, способствуют разжижении мокроты, улучшает функцию мерцательного эпителия бромгексин и муколитические препараты, Бромгексин рекомендуют детям в возрасте от 3 до б лет - в дозе 2 мг, т.е. 1/4 таблетки 3 раза в день, от 6 до 14 лет - 4 мг, т.е. 1/2 таблетки 3 раза в день. Детям в возрасте до 3 лет бромгексин не назначают! Муколитическим действием обладают нашатырно-анисовые капли и грудной эликсир (на прием столько капель, сколько лет ребенку), перкуссии (на прием от 1/2 ч.л. до 1 дес.л. 3 раза в день) и грудные сборы (№ 1: корень алтея, лист мать-и-качехи, трава душицы - 2:2:1; № 2: лист мать-и-мачехи, подорожника, корень

солодки - 4:3:3; № 3: трава шалфея, плоды аниса, сосновые почки, корень алтея, корень солодки - 2:2:2:4:4). Приготовленные отвары дают по 1/4-1/3 стакана 3 раза в день.

В стационаре с первых дней болезни назначают паровые ингаляции (детям старше 2 лет!) с отваром грудных сборов или настоями из ромашки, календулы, мяты, шалфея, зверобоя, багульника, сосновых почек (отвары готовят непосредственно перед употреблением в виде 5-10% растворов, ингаляции проводятся 3-4 раза в день). Можно использовать готовые настойки мяты, эвкалипта, кадендулы, сок подорожника, коланхоэ от 15 капель до 1-3 мл на ингаляции в зависимости от возраста. Тепловые процедуры: горчичники на грудную клетку, теплые ванны.

Диспансерное наблюдение в течение 6 мес. В целях профилактики рецидивов бронхита проводят санацию носоглотки у лиц, окружающих больного ребенка. Через 2-3 мес. назначают (детям старше 1,6-2 лет) ингаляции с отварами шалфея, ромашки или зверобоя ежедневно в течение 3-4 недель и комплекс витаминов. Профилактические прививки проводят через 1 мес. при условии полного выздоровления.

1.2. Острый обструктивный бронхит – наиболее распространенная форма острого бронхита у детей раннего возраста. Обструктивный бронхит имеет все клинические признаки острого бронхита в сочетании с бронхиальной обструкцией. Наблюдается; удлиненный выдох, экспираторный шум ("свистящий" выдох), свистящие хрипы на выдохе, участие в акте дыхания вспомогательной мускулатуры. В то же время признаки выраженной дыхательной недостаточности отсутствуют. Кашель сухой, нечастый. Температура нормальная или субфебрильная. Тяжесть состояния обусловлена дыхательными расстройствами при слабо выраженных симптомах интоксикации. Течение благоприятное. Дыхательные расстройства уменьшаются в течение 2-3 дней, свистящие хрипы выслушиваются более длительное время.

Детей раннего возраста с синдромами бронхиальной обструкции необходимо госпитализировать.

Методы обследования:

1. Общий анализ крови
2. Консультация ЛОР-специалиста
3. Аллергообследование детей после 3-летнего возраста с целью ранней диагностики бронхоспазма аллергического генеза
4. Консультация невропатолога при наличии в анамнезе перинатальной травмы ЦНС.

Лечение:

1. Эуфиллин 4-6 мг/кг в/м (разовая доза), при уменьшении симптомов бронхиальной обструкции продолжать давать эуфиллин 10-20 мг/кг в сутки равномерно каждые 2 ч. внутрь.

2. При неэффективности эуфиллина вводить 0,05% раствор алупента (орципреналина) 0,3-1 мл в/м.

3. При отсутствии эффекта и ухудшении состояния вводить преднизолон 2-3 мг/кг в/в или в/м.

В последующие дни показана спазмолитическая терапия эуфиллином тем детям, у которых первое введение препарата оказалось эффективным. Можно использовать 1-1,5% раствор этимизола в/м 1,5 мг/кг (разовая доза).

Диспансерное наблюдение заключается в предотвращении повторных эпизодов бронхиальной обструкции и рецидивов бронхита. С этой целью назначают ингаляции отваров шалфея, зверобоя, ромашки ежедневно в течение 3-4 недель в осенний, зимний и весенний сезоны года.

Профилактические прививки проводятся через 1 мес. после обструктивного бронхита при условии полного выздоровления.

1.3. Острый бронхиолит - распространенное поражение мельчайших бронхов и бронхиол, приводящее к развитии выраженной обструкции дыхательных путей с развитием симптомов дыхательной недостаточности. Болеют в основном дети первых месяцев жизни (парагриппозный и респираторно-синцитиалыш бронхиолит), но могут болеть и дети второго-третьего года жизни (аденовирусный бронхиолит).

Обструктивный синдром часто развивается внезапно, сопровождается звучным сухим кашлем. Нарастание дыхательных расстройств сопутствует резкое беспокойство ребенка, субфебрильная (при парагриппоаной и респираторно-синцитиальной инфекции) или фебрильная (при аденовирусной инфекции) температура. Тяжелое и крайне тяжелое состояние больного обусловлено дыхательной недостаточностью, Определяется вздутие грудной клетки, коробочный оттенок перкуторного звука, при аускультации легких выслушивается масса мелкопузырчатых и крепитирувщих хрипов. Диффузные изменения в легких на фоне выраженной обструкция с очень большой долей вероятности (до 90-95%) исключают пневмонию. Рентгенологически определяется вздутие легких, усиление бронхососудистого рисунка, возможны микроателектазы. Осложнениями бронхиолита могут быть рефлекторная остановка дыхания, развитие пневмонии, повторные эпизоды бронхиальной обструкции (почти у 50% больных).

Методы обследования:

1. Рентгенография легких в двух проекциях
2. Общий анализ крови
3. ЭКГ.
4. Определение кислотно-основного состояния крови (КОС)

Лечение

1. Обязательная госпитализация в стационар для оказания неотложной помощи
2. Ингаляции кислорода. Подача увлажненного кислорода через носовые катетеры, детям старше 1-1,6 лет в кислородной палатке ДПК-1 - 40% кислорода с воздухом
3. Удаление слизи из дыхательных путей

4. Инфузионная терапия в виде в/в капельных инфузий показана только с учетом гипертермии и потери жидкости при одышке
5. Антибиотикотерапия показана, поскольку в первые сутки нарастания тяжести состояния больных трудно исключить пневмонию. Назначают полусинтетические пенициллины, в частности, ампициллин 100 мг/кг в сутки в 2-3 инъекции (следует учесть, что антибиотикотерапия степень обструкции не уменьшает!)
6. Эуфиллин 4-5 мг/кг в/в или в/м (разовая доза), но не более 10 мг/кг в сутки (уменьшение тяжести обструкции наблюдается только у 50% больных!!)
7. При неэффективности эуфиллина вводить 0,05% раствор адупента (орципреналина) 0,3-0,5 мл в/м. Можно использовать ингаляции алупента 1 ил на одну ингаляцию, продолжительность ингаляции 10 мин.
8. Обструктивный синдром, длительно не купируемый введением эуфиллина, алупента, требует назначения кортикостероидов: преднизолона 2-3 мг/кг парентерально (в/в или в/м)
9. .Кардиотонические средства при тахикардии!) - в/в капельное введение 0,05% раствора коргликона 0,1-0,6 мл каждые 6-8 ч.
10. Антигистаминныв препараты не показаны! Их высушивающее, атропиноподобное действие может усилить бронхиальную обструкцию.
11. В тяжелых случаях дыхательной недостаточности назначают ИВЛ.

Диспансерное наблюдение за детьми, перенесшими бронхиолит, имеет целью предотвращение дальнейшей сенсибилизации и повторных эпизодов бронхиальной обструкции. Детям, имеющим повторные обструктивные эпизоды, в возрасте после 3 лет рекомендуется постановка кожных проб с наиболее распространенными аллергенами (пылевыми, пыльцевыми и др.).

Положительные кожные пробы, а также приступы обструкции боа вирусной инфекции указывают на развитие бронхиальной астмы.

Профилактические прививки больных, переболевшим бронхиолитом. проводят не ранее, чем через 1 мес. при условии полного выздоровления.

1.4. Рецидивирующий бронхит - бронхит, повторяющийся в течение года 3 раза и более при длительности обострения не менее 2 недель, протекающий без клинических признаков бронхоспазма, имеющий наклонность к затяжному течению. Характеризуется отсутствием необратимых, склеротического характера изменений в бронхолегочной системе. Начало заболевания может быть на первом-втором году жизни. Этот возраст имеет особое значение в возникновении рецидивов бронхита в силу слабой дифференцировки эпителия респираторного тракта и незрелости иммунной системы. Однако с уверенностью диагноз можно поставить только на третьем году жизни. Рецидивирующим бронхитом болеют в основном дети раннего и дошкольного возраста.

Клиническая картина рецидива бронхита характеризуется острым началом, повышением температуры до высоких или субфебрильных цифр. Рецидив

бронхита возможен и при нормальной температуре. Одновременно с этим появляется или усиливается кашель. Кашель имеет самый разнообразный характер. Чаще он влажный, со слизистой или слизисто-гнойной мокротой, реже сухой, грубый, приступообразный. Именно нарастающий по интенсивности кашель нередко служит поводом для обращения к врачу. Кашель может провоцироваться физической нагрузкой.

Перкуторный звук над легкими не изменен или с небольшим коробочным оттенком. Аускультативная картина рецидива бронхита разнообразна: на фоне жестковатого дыхания выслушиваются влажные крупно- и среднепузырчатые. а также и сухие хрипы, изменчивые по характеру и локализации. Хрипы обычно выслушиваются менее продолжительное время, чем жалобы на кашель. Следует обратить внимание, что у больных рецидивирующим бронхитом нередко выявляется повышенная кашлевая готовность, т.е. дети начинают кашлять после незначительного охлаждения, физической нагрузки, при очередной ОРВи.

Прогноз. При отсутствии адекватной терапии дети болеют годами, особенно заболевшие в раннем и дошкольном возрасте. Может быть трансформация рецидивирующего бронхита в астматический и бронхиальную астму. Благоприятное течение рецидивирующего бронхита наблюдается у детей, у которых он не сопровождается бронхоспазмом.

Методы обследования:

1. Анализ крови
2. Бактериологическое исследование мокроты
3. Рентгенография легких (при отсутствии рентгеновского исследования в периоды предыдущих рецидивов бронхита и при подозрении на пневмонию)
4. Бронхоскопия с целью диагностики морфологической формы эндобронхита (катаральный, катарально-гнойный, гнойный)
5. Цитологическое исследование бронхиального содержимого (мазки-отпечатки из бронхов)
6. Исследование функции внешнего дыхания; пневмотахомвтрия для определения состояния проходимости дыхательных путей, спирография для оценки вентиляционной функции легких
7. Иммунограмма

Лечение

1. Больных с обострением рецидивирующего бронхита желательно госпитализировать, но лечение возможно и в амбулаторных условиях
2. Необходимо создать оптимальный воздушный режим о температурой воздуха 18-20С и влажностью не ниже 60%
3. Антибактериальная терапия, включающая антибиотики, назначается при наличии признаков бактериального воспаления, в частности, гнойной мокроты. Курсы антибиотикотерапии (ампициллин 100 мг/кг, гентамицин 3-5 мг/кг и др.) назначаются на 7-10 дней

4. Ингаляционная терапия - один из важнейших видов терапии в лечебном комплексе, назначаемом с целью устранения нарушения бронхиальной проходимости.

Она проводится в три этапа. На первом этапе назначает ингаляции растворов солей, щелочей и минеральных вод. Эффективна для разжижения и отхождення мокроты смесь, приготовленная из равных объемов 2% раствора натрия бикарбоната и 5% раствора аскорбиновой кислоты, объем ингаляционной смеси по возрасту. При наличии слизисто-гнойной мокроты ингаляционно вводят препараты-ферменты (приложение № 1). Длительность первого этапа 7-10 дней.

На втором этапе ингаляционно вводят антисептики, фитонциды. С этой целью назначают сок лука и чеснока, отвары зверобоя (новоиманин), багульника, сосновых почек, готовые настойки мяты, эвкалипта, календулы, сок подорожника, коланхоэ, ингаляции с лизоцимом, прополисом (приложение N 2). Длительность второго этапа 7-10 дней.

На третьем этапе назначаются масляные ингаляции. Использует растительные масла, обладающие защитным эффектом. Продолжительность третьего этапа также 7-10 дней.

5. Муколитические (секретолитические) средства (см. раздел острый простой бронхит) назначают только на первом этапе проведения ингаляционной терапии
6. Отхаркивающие (секретомотормые) средства; отвары и настои трав (термопсиса, подорожника, мать-и-мачехи, чабреца, багульника, душицы), корень алтея, солодки и девясила, плоды аниса, сосновые почки. Из этих лекарственных средств составляют лечебные сборы, используемые для облегчения кашля
7. Физиотерапевтические процедуры: микроволны на грудную клетку (электромагнитные колебания сверхвысокой частоты сантиметрового диапазона, СМВ, аппарат "Луч-2" и дециметрового диапазона, ДМВ, аппарат "Ромашка".

Лечение, больных с обострением рецидивирующего бронхита проводят (в домашних условиях или в стационаре) 3-4 недели. Больные рецидивирующим бронхитом должны состоять на диспансерном учете. Наблюдение за детьми осуществляет участковые педиатр. Частота осмотров зависит от длительности заболевания и частоты рецидивов, но не менее 2-3 раз в год. Если рецидива бронхита нет в течение 2-3 лет, больного можно снять с учета. Консультации специалистов осуществляют по показаниям: пульмонолога при подозрении на развитие хронического бронхолегочного процесса: аллерголога при появлении бронхоспазма; отоларинголога для контроля за состоянием ЛОР-органов.

Реабилитацию больных рецидивирующим бронхитом проводят по принципу оздоровления часто болеющих детей:

1. Санация очагов хронической инфекции в ЛОР-органах: хронического тонзиллита, гайморита, аденоидита

2. Устранение сопутствующих заболеваний органов пищеварения: дискинезий билиарной системы, дисбактериоза кишечника и др.
3. Коррекцию метаболических нарушений назначают в течение года. Примерная схема:

- август - рибоксин и оротат калия;
- сентябрь - витамины В1, В2, пантетонат кальция и липоевая кислота;
- октябрь - настойка элеутерококка;
- ноябрь поливитаминные препараты (декамевит, аэровит, ундевит, гексавит, квадевит и др.), липоевая кислота;
- декабрь - настойка аралии, ингаляции отваром подорожника;
- январь - витамины В1, В2. пантетонат кальция и липоевая кислота;
- февраль - рибоксин и оротат калия;
- март - поливитаминные препараты;
- апрель - витамины В1, В2, пантетонат кальция, липоевая кислота;
- май - настойка элеутерококка (пантокрин).

Комплексы назначают в возрастных дозировках 10-дневными курсами
4. Препараты-адаптогены: метилурацил 0,1-0,6 внутрь 3-4 раза в день после или во время еды, 3-4 недели. Дибазол 0.003-0,03 1 раз в день. 3-4 недели
б. Ингаляции с отваром шалфея, 25-30 ингаляций ежедневно зимой, весной
6. Реаферон (генноинженерный - интерферон) интраназально в дозах 300 и 600 Т.ед в течение б дней (зимой, весной)
7. Спелеотерапия детям после 5-летнего возраста с целью нормализации мукоцилиарного клиренса и улучшения эвакуации мокроты, ежедневно, 20 сеансов
8. Лечебная физкультура
9. Массаж: точечный, классический, вибрационный
10. Закаливающие процедуры.
В период реабилитации проводится иммунологическое обследование больных. В случаях выявления синдрома иммунодефицита показана иммунокорригирующая терапия после консультации с клиническим иммунологом.
1.6. Рецидивирующий обструктивный бронхит имеет все клинические симптомы рецидивирующего бронхита, сопровождающиеся эпизодами бронхиальной обструкции. Как и рецидивирующий бронхит, относится к предастме.
Методы обследования:
Функциональная вентиляционная проба с бронходилататорами. Используются следующие показатели: жизненная емкость легких (ЖЕЛ). максимальная вентиляция легких (МВЛ), пневмотахометрия выдоха (ПТВ), форсированная жизненная емкость легких (ФЖЕЛ).
Производится запись перечисленных вентиляционных показателей до и после введения бронхолитического средства (эфедрин, эуфиллин). На наличие у обследуемых больных бронхоспазма указывает увеличение 2-3 из 4 показателей, чаще ЖЕЛ и МВЛ. Положительная функциональная вентиляционная проба с бронходилататорами, свидетельствующая о

бронхоспазме, требует проведения дифференциальной диагностики рецидивирующего обструктивного бронхита с астматическим бронхитом.
Другие методы обследования больных обструктивным рецидивирувщим бронхитом аналогичны обследованию детей с рецидивирующим бронхитом.
Лечение больных рецидивирующим обструктивным бронхитом проводят по такому же принципу, как и больных рецидивирусщим бронхитом. Дополнительно назначают препараты бронхоспазмолитики - эуфиллин, алупент (см. Лечение острого обструктивного бронхита). Диспансерное наблюдение больных имеет целью предотвращение рецидивов бронхиальной обструкции и бронхита. Реабилитация больных строится по такому же принципу, как и больных рецидивирующим бронхитом. Проведение реабилитационных мероприятий планируют с учетом результатов аллергологического обследования с наиболее распространенными аллергенами. В процессе диспансерного наблюдения и по данным аллергологического обследования диагноз "рецидивирующий обструктивный бронхит" может быть верифицирован. Вероятными диагнозами могут быть астматический бронхит, а при наличии типичных приступов удушья - бронхиальная астма.
1.6. Бронхиальная астма - хроническое аллергическое заболевание, при котором иммунопатологический процесс локализуется в бронхолегочной системе и клинически характеризуется рецидивирующими, обратимыми приступами удушья, обусловленными острым нарушением бронхиальной проходимости.
Классификация клинических форм бронхиальной астмы (С.С.Каганов, 1963)
Форма заболевания
1. Атопическая
2. Инфекционно-аллергическая
3. Смешанная
Тип
Типичная:
1. Выраженные приступы бронхиальной астмы
2. Астматический бронхит
Атипичная:
Приступы острого эмфизематозного вздутия легких
Степень тяжести
1. Легкая
2. Средней тяжести
3. Тяжелая
Показатели тяжести:
1. Частота, характер и продолжительность приступов
2. Наличие и выраженность изменений в межприступный период со стороны:
а) системы дыхания;
б) сердечно-сосудистой системы;
в) нервной системы;
г) обменных процессов:

д) физического развития;
Течение
1. С отдельными приступами, с астматическим состоянием, с асфиксическим синдромом
2. С бронхолегочной инфекцией, с воспалительными изменениями в носоглотке
3. С сопутствующими аллергическими заболеваниями:
а) с аллергическими дерматозами (экзема, крапивница, отек Квинке);
б) с другими клиническими формами респираторных аллергозов (аллергические риниты, синуиты, трахеиты, бронхиты, пневмонии, эозинофильный легочный инфильтрат)
4. С осложнениями:
а) хроническая (стойкая) эмфизема легких;
б) легочное сердце;
в) ателектаз легких;
г) пневмоторакс;
д) медиастинальная и подкожная эмфизема;
е) неврологические расстройства;
При легкой степени течения болезни обострения редкие и непродолжительные, при средне-тяжелой бронхиальной астме обострения ежемесячные. Тяжелое течение бронхиальной астмы характеризуется частыми обострениями. Приступы удушья возникает еженедельно, а нередко и ежедневно с переходом в астматическое состояние. Приступ бронхиальной астмы, длящийся от нескольких минут до нескольких часов и дней, определяется остро возникшим бронхоспазмом. Отмечается экспираторная одышка с шумным свистящим выдохом. Больных беспокоит кашель с трудно отделяемой вязкой мокротой. При перкуссии легких выявляет коробочный оттенок перкуторного звука, при аускультации множественные сухие хрипы. У детей раннего возраста в легких выслушивается разнокалиберные влажные хрипы, поскольку в этом возрасте во время приступа бронхиальной астмы преобладает не бронхоспазм, как у детей старшего возраста, а воспалительный отек слизистой оболочки бронхов и избыточная продукция слизи.
Для атопической формы бронхиальной астмы характерно острое развитие приступа и в нетяжелых случаях бронхиальную проходимость можно восстановить довольно быстро.
Обострение инфекционно-аллергической бронхиальной астмы начинается медленно и постепенно. Обструктивный синдром, при назначении бронхоспазмолитических средств купируется медленно.
В легких длительное время выслушиваются не только сухие, но и разнокалиберные влажные хрипы.
При легком приступе бронхиальной астмы самочувствие больных страдает мало. Средне-тяжелый приступ имеет клиническую картину астматического удушья. В акте дыхания участвует вспомогательная мускулатура, наблюдается тахикардия и повышение артериального давления. Тяжелый

приступ характеризуется клиническими симптомами дыхательной недостаточности на фоне выраженного астматического удушья.
Некупирующийся приступ бронхиальной астмы длительностью 6 ч. и более классифицируется как астматическое состояние, которое может перейти в астматический статус. При астматическом статусе II и III ст. наступает тотальная обструкция бронхов в результате заполнения их густым вязким секретом, выраженной воспалительной инфильтрации слизистой оболочки и спазма гладкой мускулатуры. В легких исчезают дыхательные шумы (синдром "молчания"), наблюдается снижение артериального давления, мышечная гипотония, падение сердечной деятельности.
Прогноз: течение бронхиальной астмы труднопредсказуемо. Родители больных детей не должны рассчитывать на скорое выздоровление. Их энергия должна быть направлена на проведение длительного лечения, которое позволило бы предотвратить появление новых приступов, и облегчить их тяжесть. Атопическая форма бронхиальной астмы прогностически более благоприятна при своевременном выявлении причиннозначимых аллергенов и специфической гипосенсибилизации. Инфекционно-аллергическая и смешанная формы бронхиальной астмы чаще, чем атопическая, остаются на протяжении детства, юности и становятся заболеванием взрослого.
Методы обследования:.
1. Общий анализ крови
2. Иммунограмма (определение Т- я В-лимфоцитов. Тн-хелперов, Ts-супресдоров, показателя Тн/Ts, содержания сывороточных иммуноглобулинов, циркулирующих иммунных комплексов (ЦИК'ов)
3. Исследование кислотно-основного состояния крови (КОС)
4. ЭКГ
5. Консультация ЛОР-специалиста с последующей санацией очагов хронической инфекции в ЛОР-органах
6. В межприступном периоде проведение кожных скарификационных проб с неинфекционными аллергенами.
7. Радиоаллергосорбентный тест (RAST), позволяющий выявить в сыворотке крови специфические иммуноглобулины (класса E-IgE).
Лечение
Легкий приступ бронхиальной астмы можно снять дома. В этих целях назначают бронхоспазмолитики внутрь или в виде ингаляции: эфедрин (детям от 2 до б лет по 0,003-0,01 г, от 6 до 12 лет по 0,01-0,02 г), эуфиллин 3-4 мг/кг (разовая доза) до 12-16 мг/кг в сутки. Можно использовать комбинированные препараты: теофедрин, антасман (детям от 2 до б лет 1/4-1/3 таблетки на прием, детям от 6 до 12 лет 1/2-3/4 таблетки), солутан в дозировке 1 капля на 1 год жизни. Рекомендуется также орципреналин (0,76 мг на ингаляцию или 1/4-1/2 таблетки внутрь), алупент (1-2 ингаляции или 1/4 таблетки детям до 6 лет, от 6 лет и старше 1/2 таблетки), 1,5% раствор астмопента и беротек 1-2 ингаляции, салбутамол (ингаляционная упаковка - 0,1 мг препарата, детям от 4 до 7 лет 1 ингаляция, детям школьного возраста

1-2 ингаляции), вентолин (в ингаляционных упаковках назначают в такой же дозировке, как и салбутамол, внутрь детям 3-4 лет 1/6 таблетки, 6-7 лет 1/3 таблетки, 7-14 лет 1/2 таблетки).

Больных со средне-тяжелым и тяжелым приступом бронхиальной астмы необходимо немедленно госпитализировать. В стационаре следует проводить следующие мероприятия.

Средне-тяжелый приступ купировать быстродействующими симпатомиметиками, например, парентеральным введением 0,1% раствора адреналина п/к из расчета 0,01 мг/кг в сочетании с 5% раствором эфедрина 0,6-0,75 мг/кг. Действие адреналина наступает через 15 мин, эфедрина через 45 мин., длительность действия этих препаратов 4-6 ч. Эффективно действует алупент в/м или п/к (0,3-0,5 мл), эуфиллин в/м (4-6 мг/кг разовая доза). После снятия острых проявлений средне-тяжелого приступа для стабилизации состояния больных целесообразно провести 5-7-дневный курс лечения эуфиллином или эфедрином, назначая разовую дозу препаратов внутрь 3-4 раза в день.

Противогистаминные препараты применяют, если нет затруднения при отхождении мокроты. Обязательна оксигенотерапия!

Тяжелый приступ бронхиальной астмы требует немедленного в/в введения эуфиллина из расчета 6-8 мг/кг (разовая доза) или 1 мл на год жизни, но не более 10 мл. Вне стационара препарат можно ввести струйно, но медленно, в течение 5-10 мин. в 10-15 мл 15-20% раствора глюкозы. В стационаре необходимо вводить эуфиллин в/в, капельно в 150-250 мл изотонического раствора хлорида натрия. Выраженная дыхательная недостаточность и резистентность к ранее используемым симпатомиметикам требует в/в введения преднизолона (1-2 мг/кг) или гидрокортизона (5-7 мг/кг).

Оксигенотерапия в условиях соматического стационара: увлажненный кислород по 20-30 мин. каждые 2 ч., в специализированном отделении кислородно-воздушная смесь, содержащая 35-40% кислорода.

После снятия приступа бронхиальной астмы лечение эуфиллином следует продолжить до полного устранения обструктивного синдрома, но способ введения препарата можно изменить, назначая его в/м или внутрь, или в свечах. Лечение дополняется назначением муколитических препаратов (мукалтина, бромгексина, отваров трав: чабреца, девясила, подорожника, настоев березовых почек, сосновых игл и др.).

Лечение больных с астматическим статусом I ст., представляющим собой затянувшийся тяжелый приступ бронхиальной астмы, проводят по этой же программе с дополнением антибиотикотерапии в связи с активацией бронхолегочной инфекции. Рекомендуются полусинтетические пенициллины или аминогликоаиды, возможно назначение цефалоспоринов.

При выявлении метаболического ацидоза с целью его коррекции назначают 4% раствор бикарбоната натрия из расчета 2-2,5 мл/кг под контролем pH крови (необходимый уровень 7,25); гепарин 180-200 ед/кг (под контролем коагулограммы); 1% раствор лазикса 0,5 мг/кг в сутки (при недостаточном диурезе); кардиотонические средства - 0,06% раствор коргликона детям в

возрасте от 2 до 5 лет 0,2-0,5 мл, от 6 до 12 лет 0,5-0,75 мл. Повторное капельное введение эуфиллина! Продолжать введение преднизолона, но внутрь 5-7 дней с постепенной отменой в течоние двух недель. Лечение астматического статуса проводить при назначении гипоаллергенной диеты или разгрузочного дня кефиром.

Астматический статус II ст. требует расширения объема терапевтического воздействия, направленного на восстановление бронхиальной проходимости. В этом состоянии увеличивается дозировка преднизолона до 3-5 мг/кг, который вводится в/в капельно вместе с зуфиллином. Необходима коррекция метаболического ацидоза. Клинические признаки сердечной недостаточности требуют назначения кардиотонических средств с одновременным в/в введением 50-100 мг кокарбоксилазы и препаратов калия. Показана лечебная бронхоскопия с удалением слизи и введением в просвет бронхов растворов бикарбоната натрия. По мере улучшения состояния больного доза преднизолона уменьшается до 1-1,5 мг/кг с назначением его внутрь в течение 2-2,5 недель с последующей отменой.

Астматический статус III ст. требует перевода ребенка в реанимационное отделение и назначения ИВЛ. Возможно проведение плазмофереза или гемосорбции. Дозу преднизолона увеличивают до 6-10 мг/кг, из них 4-8 мг/кг вводят в/в, 2 мг/кг внутрь. Одновременно назначают эуфиллин, кардиотонические средства по прежней программе. Лечение кортикостероидами проводят с постепенной отменой их в течение 3-4 недель. В период отмены кортикостероидов целесообразно назначение пантетоната кальция (витамин В5). витамина В6, этимизола, глицерама, индуктотермии на область надпочечников. Синдром отмены можно предотвратить назначаннием аэрозолей гормонов: бекотида, бекламата.

Контрольные вопросы

1. Автоматизированная консультативная система диагностики бронхолегочных заболеваний
2. Бронхиты
3. Классификация бронхитов

Практическая работа № 8 Автоматизированное рабочее место подготовки медико-статистических данных и АРМ врача-специалиста

Структурной единицей АСУ является автоматизированное рабочее место (АРМ) - комплекс средств вычислительной техники и программного обеспечения, располагающийся непосредственно на рабочем месте сотрудника и предназначенный для автоматизации его работы в рамках специальности. Однако простую совокупность АРМ еще нельзя считать автоматизированной системой управления. В АСУ все элементы должны быть связаны между собой средствами коммуникации (локальной сетью). Именно они, обеспечивая обмен информацией между рабочими местами, делают АСУ системой. Рассмотрим этот вопрос на примере АСУ стационара. Как известно, основным документом в стационаре является медицинская карта стационарного больного, обычно именуемая историей болезни. Именно она служит основой для объединения АРМ в систему. Речь идет об электронной автоматизированной истории болезни. Она представляет собой комплекс данных о больном, хранящихся в электронном виде в сетевой накопительной базе (в архиве электронных историй болезни).

Благодаря тому, что все АРМ связаны между собой (и, естественно, с архивом электронных историй болезни) средствами коммуникации (в данном случае – локальной сетью), каждый из компетентных сотрудников ЛПУ может работать с историей болезни любого больного непосредственно на своем рабочем месте. Так, в одно и то же время, находясь в различных помещениях, лечащий врач может записывать дневник, лаборант клинической лаборатории – вносить результаты анализа крови, а врач-рентгенолог – описывать рентгенограммы. Кроме того, средства автоматизации некоторых рабочих мест, могут автономно, без участия оператора, обращаться к историям болезни. Например, АРМ постовой сестры может выбирать из историй болезни назначения, группируя их по видам, а АРМ врача – оформлять и направлять в соответствующие службы направления на различные исследования (естественно, руководствуясь сделанными врачом назначениями).

120. Составные части асу (технические средства, информация, персонал);

требования, предъявляемые к ним. Этапы разработки и внедрения АСУ.

Основные этапы создания АСУ:

1. Предпроектное обследование объекта автоматизации с целью определения объема работ.
2. Написание технико-экономического обоснования (целесообразность создания АСУ с учетом затрат и ожидаемой эффективности).
3. Разработка технического задания (определение целей, задач, расчет эффективности АСУ, построение моделей и методов управления).
4. Разработка технического проекта:

— разработка информационного обеспечения — структура баз данных: исходная информация (карта выбывшего из стационара, талон амбулаторного

пациента и др.), справочная информация (международная классификация болезней и др.), выходная информация (макеты таблиц и т.д.);
— разработка программного и математического обеспечения;
— разработка организационного обеспечения;
— разработка технического обеспечения.

5. Ввод АСУ в действие (монтаж, наладка, обучение персонала, организация баз данных).

Основные проблемы внедрения АСУ в здравоохранении:

1. Технические проблемы (установление, наладка средств вычислительной техники, разработка программного обеспечения).
2. Организационные проблемы (организация процесса управления, создание службы по обеспечению функционирования автоматизированной системы).
3. Социально-психологические проблемы (адаптация сотрудников к новым условиям работы, их обучение и т.д.).
4. Проблемы обновления основных фондов (обновление технической базы программного обеспечения).
5. Финансовые проблемы.

Компонентами АСУ являются:

1. Технические средства – вычислительные устройства, устройства ввода-вывода, запоминающие и накопительные устройства, сетевое оборудование.
2. Программное обеспечение – компьютерные программные средства, обеспечивающие работу технических средств и обработку информации.
3. Пользователь или оператор, который осуществляет взаимосвязь с программными и аппаратными средствами системы.

Контрольные вопросы

1. Автоматизированное рабочее место подготовки медико-статистических данных и АРМ врача-специалиста
2. Требования, предъявляемые к ним.
3. Этапы разработки и внедрения АСУ.
4. Компоненты АСУ

Практическая работа № 9 Обработка данных в пакете Statistica и Excel

Пакет MS Excel насчитывает большое число пользователей. Благодаря простоте работы с ним и большому количеству полезных встроенных функций и процедур он позволяет решать многие простые прикладные задачи, связанные с обработкой данных.

Однако потребность в серьезных методах прикладной статистики и анализа данных у пользователей MS Excel при этом остается неудовлетворенной.

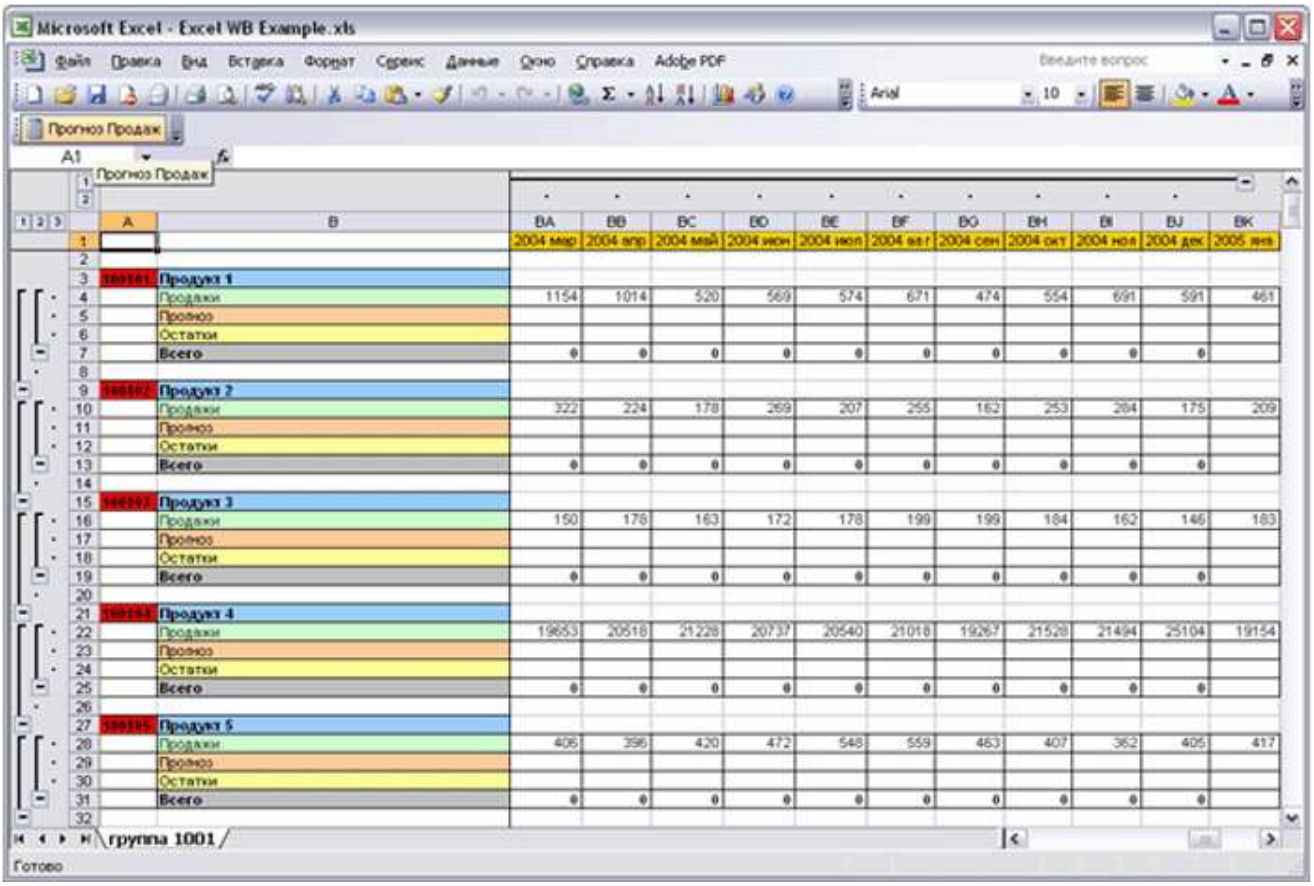
Одной из задач, часто возникающих на практике, является задача прогнозирования (например, прогнозирования продаж). Таблица ниже красноречиво говорит о весьма ограниченном наборе методов прогнозирования в Excel, по сравнению с пакетом *STATISTICA.*

метод	STATISTICA	MS Excel	комментарий
Регрессионный анализ	+	+--	В **E.** реализована лишь простейшая линейная регрессия, ограничение на число входных предикторов: не более 16. **S.** позволяет строить сложные нелинейные регрессионные модели (модули «Множественная регрессия», «Общие регрессионные модели», «Нелинейное оценивание»), при этом нет ограничений на число предикторов и их длину.
АРПСС (ARIMA)	+	-	Модель авторегрессии и проинтегрированного скользящего среднего.
Экспоненциальное сглаживание	+	+---	**E.** позволяет строить только однопараметрическую модель не учитывающую тренд и сезонность. Значение параметра задается вручную. В **S.** включено 12 базовых моделей экспоненциального сглаживания (до 4 параметров в модели) и алгоритм автоматического поиска оптимальных значений параметров модели.
Спектральный анализ	+	+--	Цель анализа - разложить комплексные временные ряды с циклическими компонентами на несколько основных синусоидальных функций с определенной длиной волн.
Сезонная декомпозиция	+	-	Цель анализа – разложить временной ряд на сезонную, тренд-циклическую и случайную компоненты.
Нейронные сети	+	-	Мощный метод моделирования, позволяющий воспроизводить чрезвычайно сложные нелинейные зависимости.
Анализ распределенных лагов	+	-	Специальный метод оценки запаздывающей зависимости между рядами.

Исходные данные из Excel можно легко импортировать в *STATISTICA*, где уже применять необходимые методы анализа. Читать подробнее

Но часто даже этой процедуры можно избежать и работать с этими методами прямо из Excel.

Рассмотрим простой пример. Имеются исторические данные о продажах ассортимента товаров некоторой компании, содержащиеся в рабочей книге Excel (см. рис. 1).

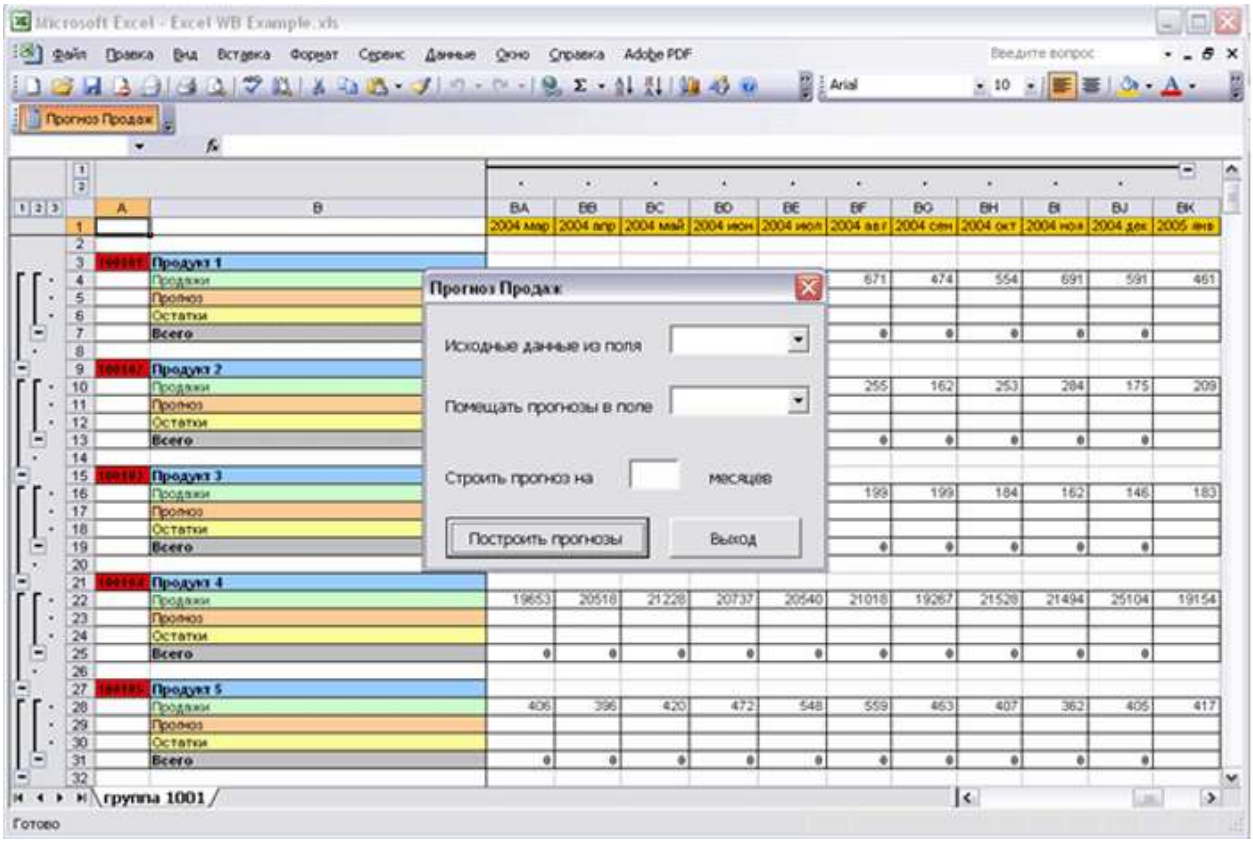


Увеличить

Рис. 1

Благодаря легкому доступу к библиотекам функций системы *STATISTICA* имеется возможность подключать мощные методы этой системы (например, методы прогнозирования) во внешние программные решения.

В данном случае путем нажатия кнопки «Прогноз Продаж» в левом верхнем углу панели инструментов Excel, можно запустить специально разработанное программное приложение для прогнозирования продаж на основе истории, накопленной в текущем файле.



Увеличить

Рис. 2

Произведя очевидные настройки в появившемся диалоговом окне

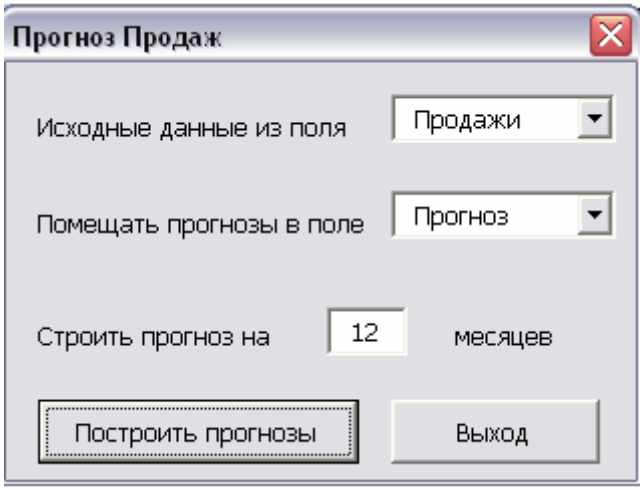
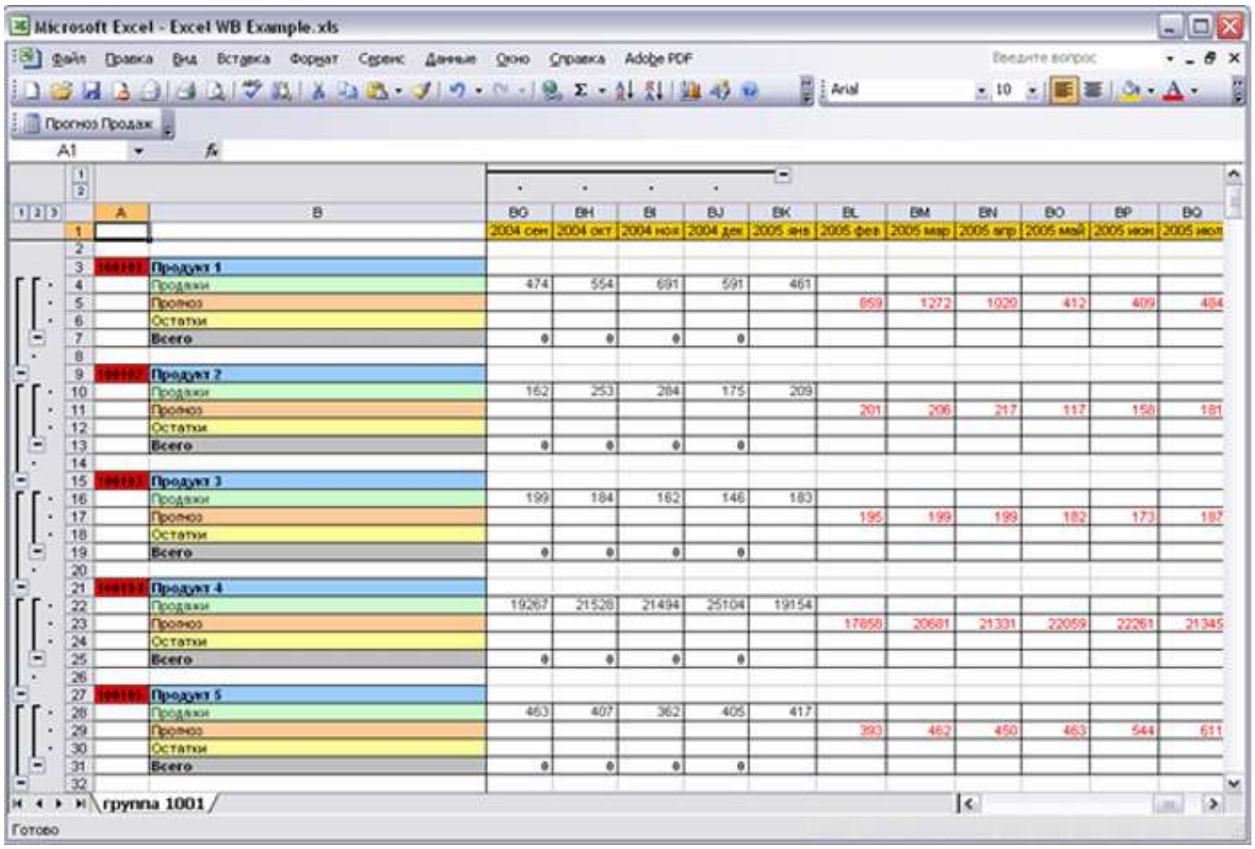


Рис. 3

и нажав кнопку «Построить прогнозы», программа в автоматическом режиме произведет подбор наилучшей модели для каждого из временных рядов продаж, содержащемся в файле, и запишет полученные прогнозы в соответствующие каждому продукту поля «Прогноз» в исходной таблице. Причем построение моделей и выбор наилучшей из них (в терминах ошибки

на тестовой выборке) будет производиться с помощью методов системы *STATISTICA*, работающей в фоновом режиме.

В итоге с помощью нескольких щелчков мыши и простейших настроек мы получаем таблицу с прогнозами на 12 месяцев по каждому из представленных продуктов (см. рис. 4).



Увеличить

Рис. 4

При необходимости использования более детального и комплексного подхода к прогнозированию продаж для решения этой задачи может применяться система «Sales-Forecast».

За более подробной информацией о прямой работе с данными из Excel и современных методах прогнозирования Вы можете обратиться к специалистам StatSoft.

Контрольные вопросы

1. Обработка данных в пакете Statistica

2. Обработка данных в пакете Excel

3. Вспомогательные редакторы.